N73-22135
NASA CR-2213

NASA CONTRACTOR REPORT

NASA CR-2213

DIGITAL PROCESSING OF RADIOGRAPHIC IMAGES

by A. D. Bond and H. K. Ramapriyan

Prepared by
COMPUTER SCIENCES CORPORATION
Huntsville, Ala.
for George C. Marshall Space Flight Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • APRIL 1973

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
NASA CR-2213		

4. TITLE AND SUBTITLE	5. REPORT DATE
Digital Processing of Radiographic Images	April 1973
	6. PERFORMING ORGANIZATION CODE M541

7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT #
A.D. Bond, H.K. Ramapriyan	

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. WORK UNIT NO.
COMPUTER SCIENCES CORPORATION Systems Development Operations, Systems Application Project Huntsville, Alabama	11. CONTRACT OR GRANT NO. NAS8-21805
	13. TYPE OF REPORT & PERIOD COVERED NASA Contractor Report - Oct. 1971 - Oct. 1972

12. SPONSORING AGENCY NAME AND ADDRESS	14. SPONSORING AGENCY CODE
National Aeronautics and Space Administration Washington, D. C. 20546	

15. SUPPLEMENTARY NOTES

16. ABSTRACT

This report presents some techniques and the software documentation for the digital enhancement of radiographs. Both image handling and image processing operations are considered. The image handling operations dealt with are, (i) conversion of format of data from packed to unpacked and vice versa, (ii) automatic extraction of image data arrays, (iii) transposition and 90° rotations of large data arrays, (iv) translation of data arrays for registration and (v) reduction of the dimensions of data arrays by integral factors.

Both the frequency and the spatial domain approaches are presented for the design and implementation of the image processing operations. It is shown that spatial domain recursive implementation of filters is much faster than nonrecursive implementations using Fast Fourier Transforms (FFT) for the cases of interest in this work. The recursive implementation of a class of matched filters for enhancing image signal to noise ratio is described.

Test patterns are used to illustrate the filtering operations. The application of the techniques to radiographic images of metallic structures is demonstrated through several examples.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Computer Image Processing Digital Filtering Structural Radiography Pattern Recognition	

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
UNCLASSIFIED	UNCLASSIFIED	208	$3.00

ACKNOWLEDGEMENT

The authors wish to acknowledge the encouragement and support provided by Mr. B.C. Hodges, Branch Chief, and Mr. D.T. Thomas, Project Technical Monitor, Applications Development Branch, Computer Systems and Simulation Division, Computation Laboratory, Marshall Space Flight Center. The authors also appreciate the helpful contributions and criticisms provided by Mr. Thomas.

TABLE OF CONTENTS

PART I

PART II

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

PART I

LIST OF ILLUSTRATIONS (Continued)

PART I (Continued)

LIST OF ILLUSTRATIONS (Continued)

PART II

DIGITAL PROCESSING OF RADIOGRAPHIC IMAGES

SUMMARY

In the nondestructive testing of metallic structures using radiographs it is sometimes necessary to enhance the images to aid visual interpretation. The digital computer is a very useful tool for this purpose, providing a means to the accurate and flexible implementation of well known filtering techniques and to the development of new techniques.

This report provides a description of some of the techniques for image handling and image processing on the computer. Both the frequency and spatial domain designs and implementations of digital filters are covered. Examples of digitally processed synthetic test patterns as well as radiographs are shown to illustrate the techniques. The software developed implementing the processing algorithms is documented.

It is seen that digital processing enhances the radiographic images quite noticably, but noise in the radiographs is a limiting factor. Further work needs to be done in the application of feature-oriented image enhancement and pattern recognition methods to radiographs.

SECTION I. INTRODUCTION

A. Radiography and the Digital Computer

The basic problem considered in this report is that of facilitating the identification of internal flaws in metals by using digital computer techniques of image enhancement on the radiographs of these metals.

Conventionally, the hidden flaws in metals such as blow holes in a welded joint and hairline cracks are detected visually by looking at dense radiographs of the suspected areas under a bright light with a magnifying glass. While many of the flaws can be identified in this manner, some cases arise in which the existence of flaws determined visually becomes questionable. In such cases it is necessary to enhance the radiographs by some method to confirm the existence of flaws. A typical example of such a problem is one in which the beginning of a crack is visible in the radiograph, but the end of the crack is very thin and merges into the background of the radiograph so that it is difficult to find how far the crack really extends.

A large variety of techniques exists for image enhancement. These techniques can be grouped broadly into two classes, viz., optical and digital. Optical methods consist of preparing transpariences which represent desired filtering operations and producing images by passing coherent light (or non-coherent light in some techniques) through the given picture and the filter transparancies. These methods handle all points in a given picture at one time and hence are virtually instantaneous. They are, however, restricted mainly to linear operations and their accuracy is limited by the accuracy with which physical filters (transparencies or apertures) can be made. On the other hand, digital techniques are very flexible and accurate. They can be used for nonlinear operations also in a controlled manner. In fact some operations such as adaptive filtering where the filter characteristics change according to the data content of the part of the image being processed are impossible using optical methods exclusively, are extremely cumbersome using electro-optic methods, but can be performed digitally with comparative ease. The main disadvantage of digital methods is that they handle the picture data sequentially and therefore they are slow. But, their accuracy and flexibility render the digital computer a very useful tool at least during the technique development stage when one wants to explore various processes of image enhancement. Many of the techniques so developed can in fact be implemented optically for fast processing on a production basis.

We shall concentrate here on the digital techniques since at the present stage of learning and developing enhancement methods for radiographic images it is necessary to use digital methods.

B. Definition of Digital Processing of Pictures

1. Picture Function. A "picture" can be mathematically described by a real function S (x, y) of two real variables x and y which gives the "gray level" of the picture at the point (x, y). Figure 1.1 shows a picture whose picture function is given by

$$S(x, y) = 1 \quad \text{for} \quad \begin{array}{l} (0 \leq x \leq .2,\ 0 \leq y \leq .5) \\ (x = .5,\ 0 \leq y \leq .5) \\ (.8 \leq x \leq 1,\ 0 \leq y \leq .5) \\ (0 \leq x \leq 1,\ .5 \leq y \leq 1) \end{array}$$

and

$$S(x, y) = 0 \quad \text{elsewhere}$$

2. Noise and Filtering. In the process of producing a picture there are several factors besides the true image itself that contribute to the final picture function. Some of the factors are [1]

(a) Random noise in the film

(b) Background of the object surrounding the desired part of the object

(c) Fluctuations in the light output of the image scanning system

(d) Electrical noise in the output of the light sensing device in the scanner

(e) Finite aperture size of the scanner

All the contributions to the picture function other than due to the true image (or the desirable part of the true image) will be termed noise. If noise is additive and the "picture function" of noise is $\nu(x, y)$, then the actual picture function obtained will be

$$S_1(x, y) = S(x, y) + \nu(x, y) \tag{1.1}$$

When the desired picture function is S (x, y), in general the actual picture function obtained can be written as

$$S_1(x, y) = f(S(x, y),\ \nu(x, y)) \tag{1.2}$$

The process of improving the quality of the image consists of filtering out the noise $\nu(x, y)$ so that S (x, y) can be recovered from S_1 (x, y).

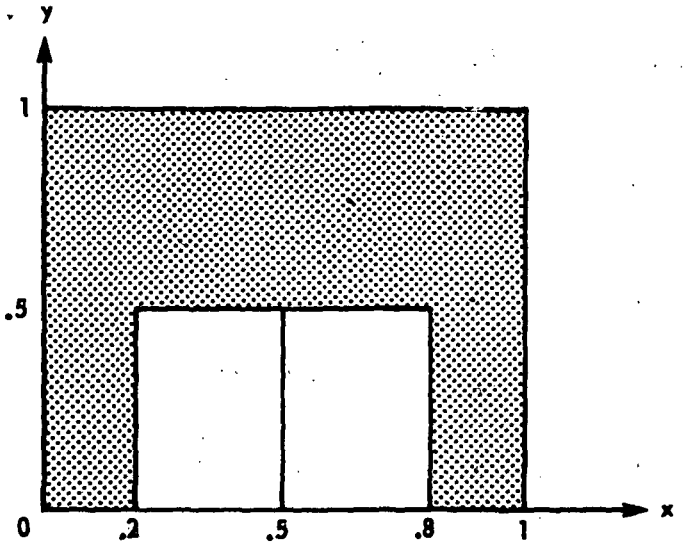


FIGURE 1.1a A SAMPLE PICTURE.

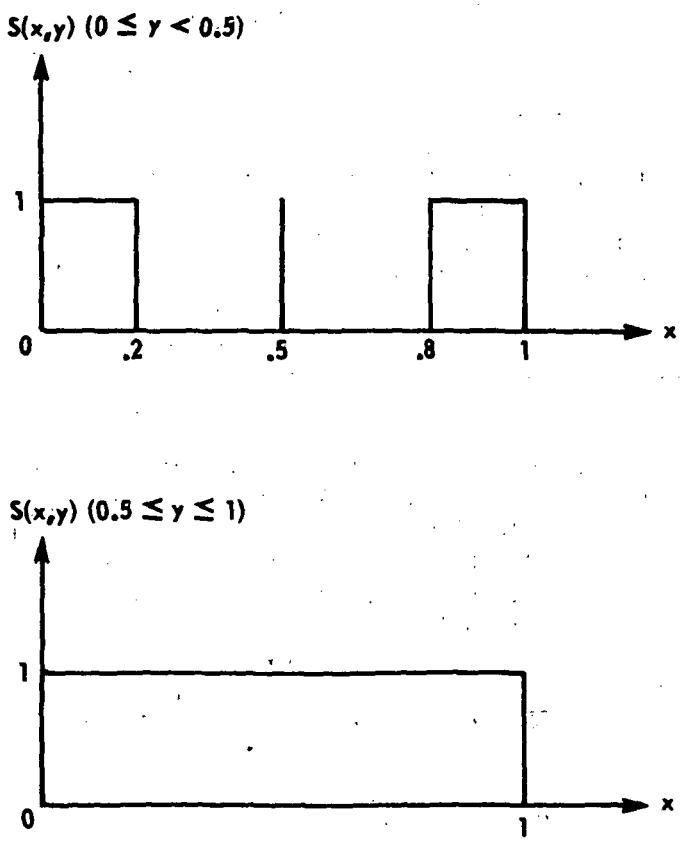


FIGURE 1.1b. THE PICTURE FUNCTION FOR THE PICTURE IN FIGURE 1a.

3. Digital Filtering. A digital computer can be used as an efficient tool in picture processing by using the well known techniques of digital filtering [2]. In order to do this, it is first necessary to obtain image data in digital form. This is done by scanning the picture with a scanning microdensitometer, sampling and digitizing the densitometer records, and thereby producing a sequence $\{S_1 * (i, j)\}$ i=1,..., M and j=1, ..., N where

$$S_1 * (i, j) = S_1 (x_i, y_j) \tag{1.3}$$

and (x_i, y_j) are the points at which the picture function S (x, y) is measured by the scanner. A digital filter is just a transformation of the sequence $\{S_1 * (i, j)\}$ which produces a new sequence $\{S_2 * (i, j)\}$. The sequence $\{S_2 * (i, j)\}$ can be converted back to a picture, for example, on a CRT display unit.

The above steps in picture processing are summarized in Figure 1.2.

The philosophy of software development is governed to a great extent by the hardware available for implementing it. Hence, a brief description of the facilities which represent the three main blocks of Figure 1.2 is given in the following subsection.

C. Description of the Processing Facilities

The hardware used for the digital processing applications discussed in this report is described below. The microdensitometer shown in Figure 1.2 is an Optronics Photoscan P-1000. It is a rotating drum scanner whose apertures and scanning intervals can be set to 12.5μ , 25μ and 50μ . It can be used to quantize densities between 0 and 2D or 0 and 3D in equal steps either as eight bit (0 to 255) or six bit (0 to 63) integers. The range of densities can be increased beyond 3D by using a neutral density filter as bias.

The microdensitometer is currently being used on its six bit mode which is convenient for use on the IBM 7094 (with 7-track tapes and 36 bit words) on which all the digital processing is implemented. The scanner generates six bit numbers on tape and hence each 36 bit word corresponds to the densities at six neighboring picture points (pixels) on a scan line. Since the IBM 7094 is a word oriented machine it is necessary to unpack each of the 36 bit words generated by the microdensitometer into 6 words before performing any arithmetic operations on it. Also, at present, the IBM 7094 has only tape units and hence all the software is geared towards using work tapes (not disk files).

There are two choices for the picture reconstruction unit in Figure 1.2, viz., a DICOMED 31 storage CRT display unit and an Optronics Photowrite P-1500. The DICOMED 31 is a 64 grey level display unit which converts each 36 bit word on packed tape into six neighboring pixels on a horizontal line on the screen of a storage CRT, the intensity of the

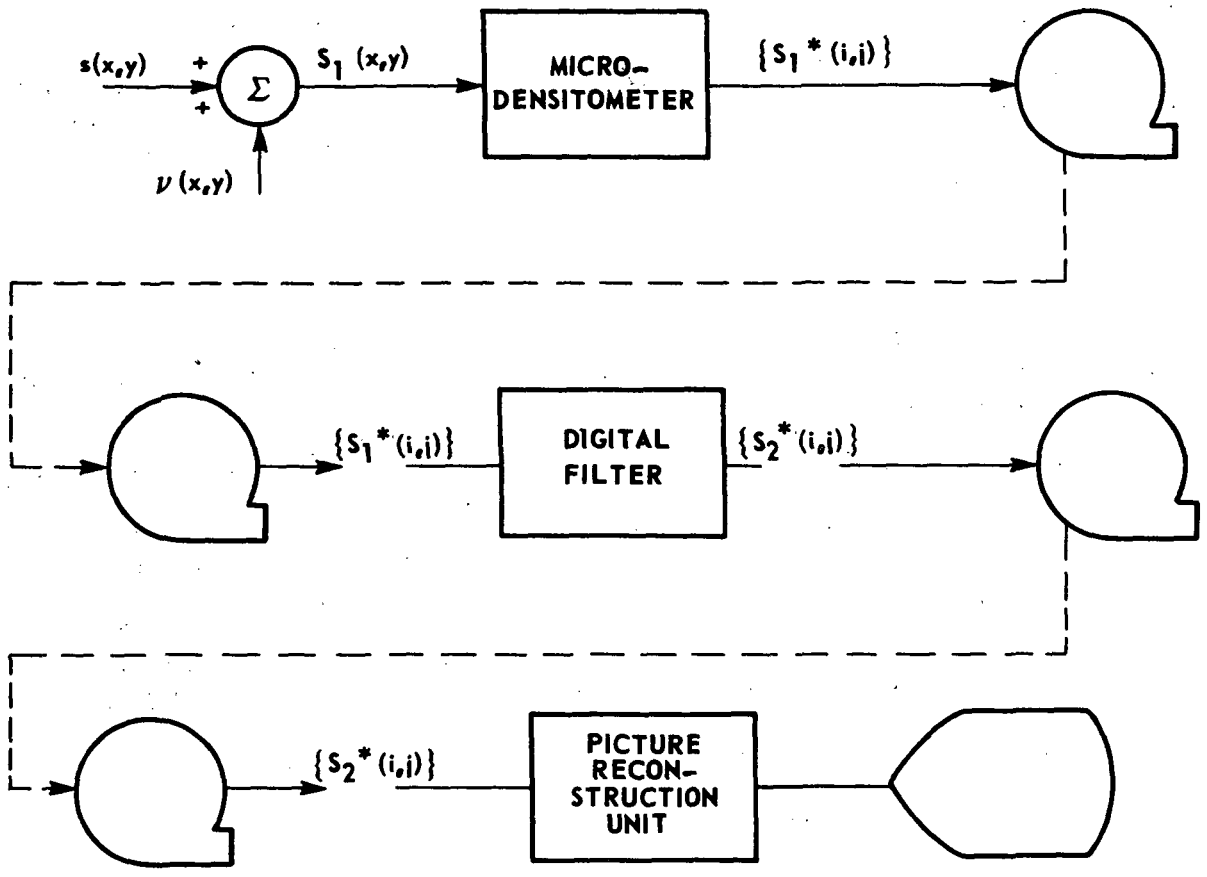


FIGURE 1.2 STEPS IN PICTURE PROCESSING.

pixel being proportional to the corresponding 6 bit number. It generates one horizontal line per record. The screen capacity is 1024 x 1024 pixels. It is required that each record on tape should have at least 171 words (i.e., 1026 pixels). If a record has more than 171 words all but the first 171 words of the record will be ignored. The writing on the screen is stopped when 1024 records are written or an end of file mark is encountered on the tape. The Photowrite is a rotating drum device which converts digital data into a picture on a film. The writing aperture and interval can be set to 12.5μ , 25μ and 50μ . It can convert either 6 bit and 8 bit numbers into pixel density on film. It is currently being used in its 6 bit mode to be compatible with the rest of the equipment.

The rest of this report presents the techniques and software which have been developed for digital image enhancement using the above facilities and shows examples illustrating the application of these techniques to radiographic images.

D. Report Outline

This report is presented in two main parts. The first part is mainly a description of the techniques and the second part is a documentation of the software developed.

The techniques and software are classified for convenience as Image Handling and Image Processing. The term Image Handling is used to cover such operations as unpacking and packing of the data, geometrical operations on image data such as transposition of arrays or rotations and translations. The term Image Processing refers to modification of image data arrays using point or local operations. Section II considers Image Handling operations for which software was developed, as and when the particular operations were found necessary. Section III describes the image processing operations, considering in particular techniques for the design and implementation of digital filters both in the spatial frequency domain and the spatial domain. Section IV presents the results of the processing operations applied to sample radiographs.

SECTION II. IMAGE HANDLING OPERATIONS

In the course of developing techniques and software for radiographic image processing, it was found necessary to produce software for various image handling operations. The software, while designed mainly for radiographic image processing, includes in some cases, features which are useful for other purposes also where the inclusion of these features did not involve much additional labor.

A. Automatic Picture Extraction

The first step one has to go through in processing picture data after a tape of digital data is generated by the microdensitometer is to convert the data into a form suitable for handling on the computer. Also, in the interest of minimizing processing time it is desirable to extract as small a rectangular array of the digitized data as possible containing the region of interest in the picture. Further, as noted in section I-C it is necessary to unpack each word of digitized data into six words before performing arithmetic operations on IBM 7094.

Manually, one can display the data on the tape from the microdensitometer on the storage CRT screen and superimpose a grid on it and then determine the coordinate bounds of the region of interest relative to the top left corner of the screen. This is a useful procedure when the picture has sufficient contrast so that some features in the region of interest can be easily identified from the display on the screen. However, in the case of most of the structural radiographs, it has been found that when the original digitized version is displayed on the screen the contrast is so low that it is difficult to identify the region of interest. Hence the following technique has been developed to determine the region of interest and extract it in unpacked form onto a tape. The technique will be first described and the assumptions needed and its limitations are discussed later.

In this technique, the first step is to mark the area of interest by masking the remaining areas of the radiograph with an opaque material so that the area of interest is an approximately rectangular window. Then, the radiograph is digitized so that the masked regions of it are included in the scan on all sides of the region of interest. Next, each record of the digitized data is read and examined for points with density numbers less than 63 (i.e., the highest density number with 6 bit digitization). The first record which has density numbers less than 63 is noted. Starting with this record the points A_i and B_i are determined for the ith record as shown in Figure 2.1. The point A_i is the first point in the ith record which has density number less than 63 and the point B_i is the last point in the ith record whose density number is less than 63. Also, the last record of interest is determine by now finding the first record all of whose density numbers are equal to 63. In the ideal case where the mask has an exactly rectangular window whose sides are parallel and perpendicular to the scanning direction, and where all the density numbers

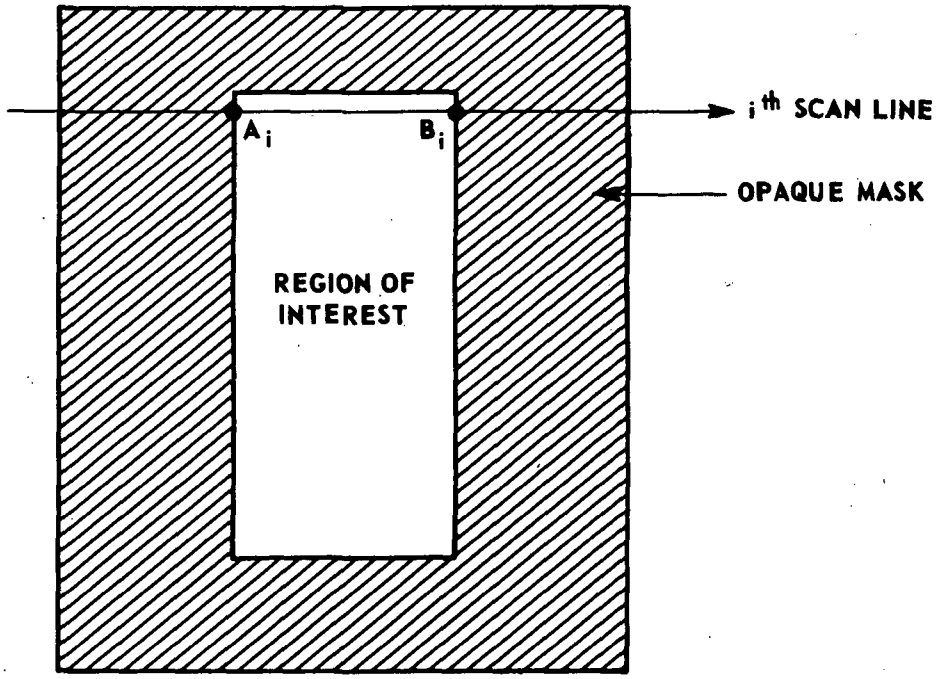


FIGURE 2.1 AUTOMATIC PICTURE EXTRACTION

in the region of interest are less than 63, this method will yield the exact boundary of the region of interest and the pixel numbers of A_i and B_i in the ith record will be constant with respect to i. If however, it is known that most of the density numbers in the region of interest are less than 63 and that there are no long sequences of consecutive points with density numbers equal to 63 along a scan line at either end of the region (i.e., no highly dense straight lines in the scanning direction) then this method will yield the locations of A_i and B_i in the various records very close to, but not exactly on, the beginning and end of the boundary of the window in the mask. Further, margins can be allowed on all four sides to compensate for the window in the mask not being exactly a rectangle and for possible skewness in its orientation with respect to the scanning direction. Then, to ensure that the area which is finally extracted is within the window marked by the mask, the right most location of A_i and the left most location of B_i are chosen after allowing the specified vertical and horizontal margins. Thus the area to be extracted is determined. After finding the area to be extracted the data tape is rewound and the required part of the data is unpacked and written on another tape.

This method works under the following assumptions:

(a) The region of interest can be isolated using an approximately rectangular window in a mask.

(b) The skewness in the sides of the mask is limited such that the right most location of A_i is to the left of the left most location of B_i (after vertical and horizontal margins are allowed).

(c) The region of interest does not contain long dense strips in the scanning direction which give density numbers equal to 63 at either end of the region.

It has been found that in most of the radiographic work done so far, the above assumptions are satisfied. A margin of 20 to 30 pixels on all sides has been used successfully in extracting the region of interest in several radiographs digitized with a 12.5μ scanning interval.

The details of the program APES (Automatic Picture Extraction and Scaling) which implements the above technique are presented in Part II of this report. This program includes, as options, linear stretching of density numbers between 0 and 63 on the basis of the maximum and minimum density numbers in the region of interest and overriding the automatic determination of the region to be extracted and extracting an externally specified rectangular region.

B. Transposition and 90° Rotations

In many picture processing applications, it is necessary to perform filtering operations in two dimensions. Several two dimensional operations on pictures lend themselves to implementation as two one dimensional operations either in tandem or in parallel. Such operations can be referred to

as separable operations and are much simpler to implement than general two dimensional operations which are not separable.

One dimensional operations handle one record of data at a time. Therefore, a horizontal filtering operation (i.e., along the scanning direction) on digitized picture data can be performed directly from the tape generated by the picture extraction software described in section II-A. However, to perform vertical filtering (i.e., perpendicular to the scanning direction) it is necessary to have one point from each of the records on tape. Therefore to simplify data handling it is necessary to transpose (or rotate through 90°) the data array so that pixels along a straight line perpendicular to the scanning direction will form a record on tape. Since the data arrays are almost always too large to fit in core, it is necessary to use a special technique to transpose (or rotate) the data arrays. This technique is described below.

This technique treats the given matrix A as made up of several 128 x 128 submatrices. If the number of rows or columns of A is not a multiple of 128, the matrix A is padded at the ends with arbitrary numbers which are immaterial so that it can be treated as a composite of 128 x 128 submatrices. Denote the 128 x 128 submatrices of A (padded if necessary) by A_{11}, A_{12}, ..., A_{1n}, ..., ..., A_{m1}, ..., A_{mn}, as shown in Figure 2.2. Also, let A_1, A_2, ..., A_n denote the submatrices of A consisting of the first, second, ..., n^{th} 128 columns of A.

Step 1: The program reads the input matrix A row by row and splits each row into n parts and writes each part on a work tape. Thus, at the end of one reading of the input tape, the i^{th} work tape will have the submatrix A_i for i = 1, ..., n where each row of the matrix is written as one record. The input tape and work tapes are rewound.

Step 2: The submatrix A_{11} is read from the first work tape (i.e., the first 128 records on the tape) and A_{11}^T is written as one logical record on the output tape. A_{12} is read from the second work tape and A_{12}^T is written as the second logical record on the output tape. Thus, at the end of this step the output tape will contain mn records, in the order A_{11}^T, A_{12}^T, ..., A_{1n}^T, ..., ..., A_{m1}^T, ..., A_{mn}^T. The output tape and the work tapes are rewound.

Step 3: Now, A_{11}^T is read from the output tape as one logical record and written on the first work tape as 128 logical records. Similarly, A_{12}^T, ..., A_{1n}^T are written on the first work tape. Thus, A_{i1}^T, ..., A_{in}^T are written on the i^{th} work tape for i = 1, ..., m. The work tapes and the output tape are rewound.

Step 4: Now, note that the first logical record on the i^{th} work tape is the first row of A_{i1}^T. Therefore, the first record is read from each of the work tapes, and the first row of A^T is formed and written on the output tape as one logical record, remembering to throw away the arbitrary numbers, if any, in A_{m1}^T which were used for padding. Thus, all the N rows of A^T are written on the output tape as one logical record each.

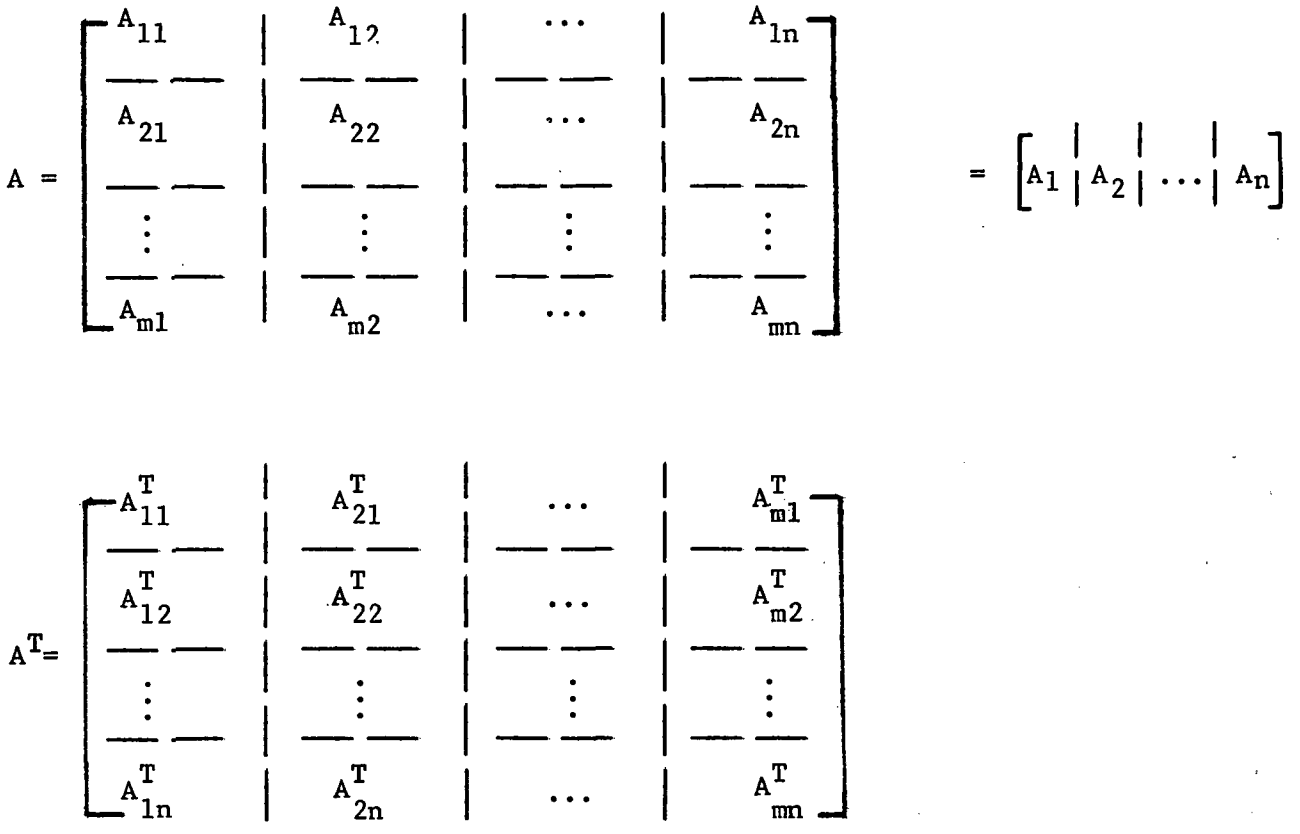


Figure 2.2. The Partitions of the Matrix A and its Transpose

The above method needs only a very small modification in order to generate 90° rotations instead of the transpose of the data array.

First of all, note that if $\hat{A}$ denotes the rotation of an M x N matrix A about its center by 90° in the counterclockwise direction, then

$$\hat{a}_{ij} = a_{j,N-i+1} \qquad \text{for} \qquad \begin{matrix} i=1,\ldots,N \\ j=1,\ldots,M \end{matrix} \tag{2.1}$$

where x_{ij} denotes the i^{th} row, j^{th} column element of the matrix X.

Also, if $\check{A}$ denotes the rotation of A about its center by 90° in the clockwise direction, then it follows that

$$\check{a}_{ij} = a_{M-j+1,i} \qquad \text{for} \qquad \begin{matrix} i=1,\ldots,N \\ j=1,\ldots,M \end{matrix} \tag{2.2}$$

Therefore, it is obvious that

$$\hat{a}_{ij} = a^T_{N-i+1,j} \tag{2.3a}$$

and

$$\check{a}_{ij} == a^T_{i,\ M-j+1} \tag{2.3b}$$

for i=1,...,N and j=1,...,M. Hence, to generate $\hat{A}$ the first step is modified so that when each row of A is read, the elements are first reflected about the center of the row and then the record is split into n parts and written on work tapes. To generate X, the last step is modified so that each record of A^T is reflected about the center of the record just before writing it on the output tape.

C. Reflection

In many filtering operations it is necessary to find the convolution of an array of filter weights with the data array. If the data array is an M x N matrix A and the filter array is an m x n matrix G, the output array B is given by

$$b_{ij} = \sum_{r=1}^{m} \sum_{s=1}^{n} g_{rs} \ a_{i-r,\ j-s} \tag{2.4}$$

It is seen that the formula (2.4) gives only a (M-m) x (N-n) array B if only the points a_{ij} in the array A with subscripts i=1,...,M and j=1, ...,N are to be used. In order to get a filtered picture of the same size as the input picture it is necessary to define a_{ij} outside the given range i=1,...,M and j=1,...,N. The definition of a_{ij} outside the range determines the values of b_{ij} in equation (2.4) for i=1,...,m and j=1,...,n. Even though a_{ij} can be defined arbitrarily outside the range of the input array it has been found experimentally that in most cases the edges of the output picture will appear better if a_{ij} are defined for points outside the range by reflecting the given picture array in the corresponding edges.

That is, for example,

$$a_{ij} = a_{2-i,j} \qquad \text{for} \qquad 2-M < i < 1 \tag{2.5a}$$

$$\text{and} \quad a_{ij} = a_{i,g-j} \qquad \text{for} \qquad 2-N < j < 1 \tag{2.5b}$$

Now, horizontal filtering operations require handling the data sequentially record by record, each record being generally small enough to be held in core. In these cases, the required reflections in vertical edges can be easily generated in core during computation. But in computations where reflections of data in the horizontal edges of the picture are needed, as in two dimensional fast recursive filters discussed in Section III, it is necessary to generate data files containing the reflections of the data array in the first and the last records. The number of records to be reflected depends upon the number of rows in the filter array. The method for generating the reflections when the part of the data array to be reflected does not fit in core is outlined below. It is assumed that M_r records are to be reflected in both the top and bottom edges of the array and M is the total number of input records.

Step 1: The dimensions of a two dimensional array W are specified according to the number of words per record in the given data array and the core capacity, the purpose being to store as many records of data as possible in core. Suppose the number of rows in W is m. Then the number of work tapes needed is the highest integer in $(M_r-1)/m$. This number is found. The next steps are described in relation to figure 2.3 where the number of work tapes is assumed to be 2.

Step 2: The first m records of data starting with the second record (constituting the subarray A_1 of the input data) are read into core and transferred to the first work tape. Similarly the subarray A_2 is transferred to the second work tape. Now, the remaining (M_r-2m) records constituting the subarray A_3 are read into core and written on the output file in reverse order (subarray $\bar{A}_3$). Next the array A_2 is read from the second work tape and written on the output file in reverse order (array $\bar{A}_2$). Next, A_1 is read from the first work tape and written on the output file in reverse order (array $\bar{A}_1$).

Step 3: The input file is now rewound to the beginning of the file and the first $(M-M_r-1)$ input records are copied onto the output file.

Step 4: The next m records of input constitute the subarray A_5. These are read into core and copied onto both the first work tape and the output file. Similarly, the subarray A_6 consisting of m records is copied onto both the second worktape and the output file. Now, the next (M_r-2m) records are read into core and the array A_7 is formed. The last record of input is now copied to the output file and the arrays A_7, A_6 and A_5 are written in reverse order on the output file as $\bar{A}_7$, $\bar{A}_6$ and $\bar{A}_5$ respectively.

D. Packing and Centering for Image Reconstruction

After the processing operations are performed on the image data and the resulting numbers are converted into integers between 0 and 63

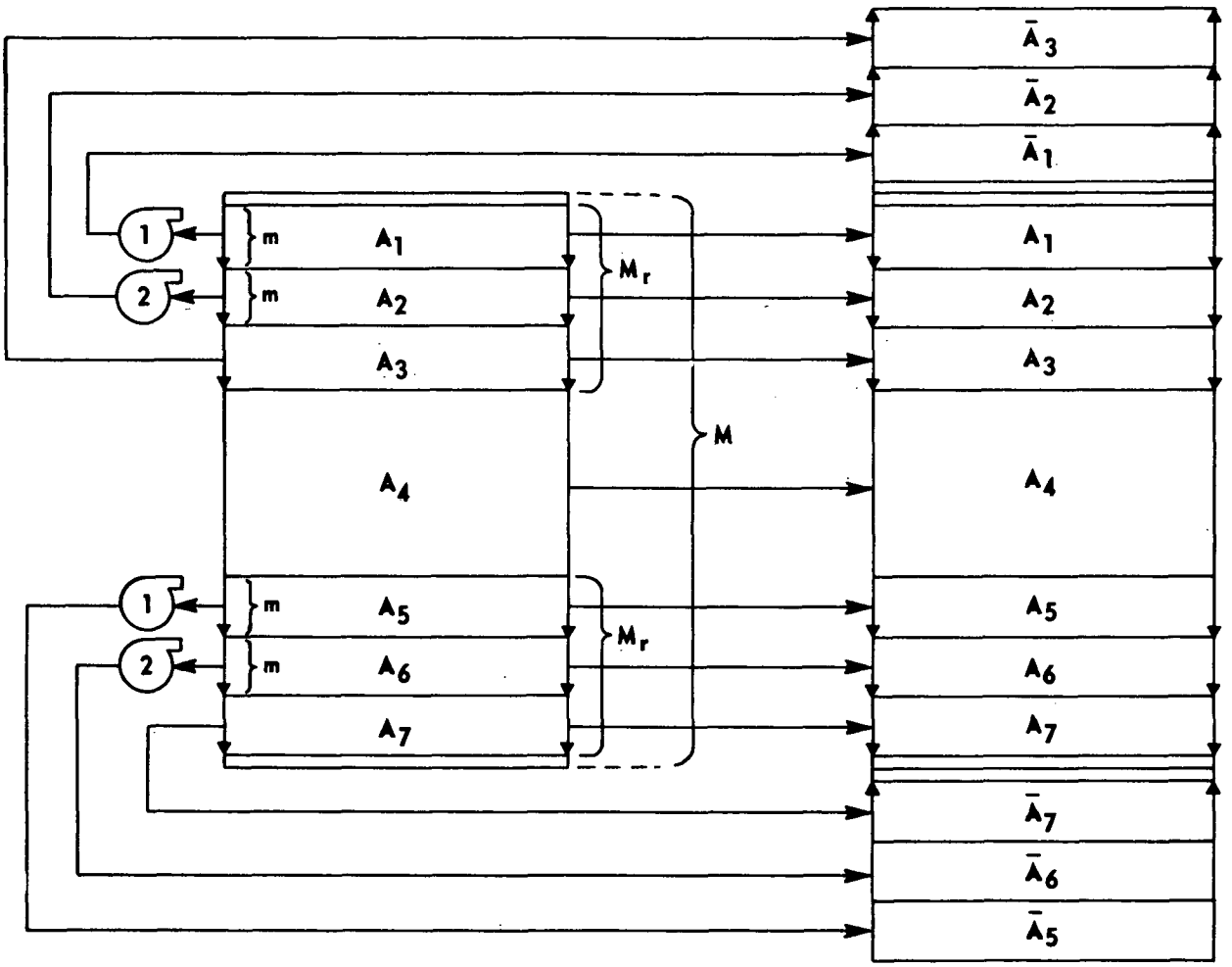


FIGURE 2.3 REFLECTIONS OF DATA ARRAY A.

(by some type of scaling if necessary), the data must be packed so that it is in a form required for the image reconstruction devices. Also, it is desirable to arrange the data so that the image when reconstructed is properly centered on the screen or film. Further, in many cases where it is necessary to study the effects of different filtering operations on a picture it is useful to display several images on the screen at a time. Therefore, a program has been developed which will prepare data for multiple frame display (or multiple frame writing on film). This program can also enlarge a picture by specified integral factors in vertical and horizontal directions by repetition of pixels. A brief description of its operation is given below.

It is assumed that the number of frames to be packed is a composite number N and its factors, viz., the number of frames in the vertical (N_v) and horizontal (N_h) directions are specified. Also, the enlargement factors are assumed to be given. In the case of preparing data for display on the screen the screen size (in the present set up) is known to be 1024 x 1024 pixels. In the case of preparing data for writing pictures on film it is assumed that the film width and interval desired between vertical frames is specified in millimeters and the writing interval (i.e., interval between pixels) is specified in microns. The program first computes the horizontal and vertical margins between frames in terms of number of pixels. Next, if $N_h > 1$, the $N_h \times N_v$ frames are packed and written on N_h work tapes, the i^{th} work tape containing the i^{th} set of N_v consecutive pictures, repeating pixels if enlargement is required and the areas corresponding to margins being filled with zeros. The i^{th} record to be written on the output tape is formed by combining the i^{th} records on each of the work tapes, for i=1,2,3,... .

E. Other Image Handling Operations

There are several other image handling operations which have been found useful. These operations are very simple to perform and do not warrant a formal description of techniques. Some of these operations and their purposes are discussed below.

1. Translation of Pictures. The term translation here refers to shifting of picture data by specified amounts in the horizontal and vertical directions. This is essentially a re-indexing operation and is useful for registration of two images. It is well known that point by point averaging of several pictures of an object reduces the random noise content of the image and hence improves the signal to noise ratio. But in performing the averaging digitally, it is necessary to align (register) the data so that the same point on the various pictures of a scene has the same address in the digitized files of these pictures. Therefore, a program has been developed for translating the picture data to obtain proper registration given the amount of shift needed.

2. Reduction of Dimensions of a Picture. When an area of a picture is digitized with high resolution it is possible that all of the image cannot be displayed on the screen. Also, it may be desirable, sometimes, to work on an image first with a reduced resolution so that the

entire image can be processed quickly and an area of interest can be identified for further processing. Therefore, a program has been developed for reducing the dimensions of a given picture data file by given integral factors either by skipping points or by averaging density numbers over rectangular blocks of points.

SECTION III. IMAGE PROCESSING OPERATIONS

This section presents the mathematical details of the image processing software developed. This section is divided into two main subsections. The first subsection considers the filter design techniques from either frequency or spatial domain specifications. The second subsection considers filter implementation techniques.

A. Filter Design Techniques

In picture processing it is necessary to take into account the nature of the object whose image is being processed, and also the nature of the corrupting noise which might be due to various sources such as fluctuations in the light source and the electrical noise in the output of the light sensing device in the image scanning system. A useful technique for describing the nature of the object and the various noises and modifications introduced by the imaging system is the "frequency domain" method which is based on the Fourier Transform. This method is useful when all the systems and operations under consideration can be assumed to be linear and spatially invariant and the noise can be assumed to be stationary. When such assumptions are not valid one has to design and implement the filtering operations indirectly in the spatial domain. Both the frequency and spatial domain methods of filter design are discussed below.

1. Frequency Domain Approach: In this method, as in classical Control Theory and Network Analysis, the inputs (i.e. the spatial variations of the intensity of the light from an object) to the imaging system, the outputs from the imaging system (i.e. the spatial variations of the intensities recorded on the film) and the imaging system itself are described in the Fourier transform domain. In other words, the inputs and outputs are decomposed into sinusoidal components of varying frequency amplitude and phase and the imaging system is described by its "Optical Transfer Function" (OTF) which is the relative amplitude and phase response of the system to sinusoidal inputs of varying frequency. If the imaging system is linear, it is well known that the output spectrum (i.e. the Fourier transform of the output) is just the product of the OTF of the system and the input spectrum. [3]

Also, if the input spectrum is band limited then the output spectrum is also band limited. Thus, if we are interested in finite resolution (as is most commonly the case) then the spectrum of the pictures produced by the imaging system would be band limited. Then, if the picture is digitized using a sampling rate greater than twice the maximum frequency present, the spectrum of the digitized picture is the same as that of the original picture over the frequency range of interest. Therefore, even the digital components of the image processing system such as the digital filter can be described using the frequency domain approach, and the output spectrum of any component is the product of the input spectrum and the transfer function of the component.

In order to apply the concepts of frequency domain analysis used in control theory to the analysis and design of image processing systems, it is necessary to use a two dimensional Fourier transform instead of the usual one dimensional Fourier transform since there are two independent (spatial) variables (x, y) in place of one independent (time) variable t. For simplicity, the analysis will be carried out using one dimensional Fourier transform and the analysis can be easily extended to two dimensional filtering.

a. Choice of filter transfer function: The first step in designing a digital filter in frequency domain is to obtain the specifications of the filter. In the frequency domain, specifying a filter consists in giving the desired range of frequencies that are to be passed unchanged and the range of frequencies that are to be stopped, attenuated or amplified. For example, a hairline crack in a metal or a bone is a high frequency phenomenon compared to the rest of the object and hence if the crack is to be made prominent the low frequencies must be suppresssed. Thus, a high pass filter should be used. Some of the other classes of filters are low pass, band pass and high emphasis.

The magnitude of an ideal low pass filter transfer function is shown in figure 3.1a. The transmission in the pass band is unity and that in the stop band is zero. There is a discontinuity at the transition between the bands. Transfer functions can be digitally realized with fewer terms (and hence yield faster filter implementations) if they are continuous functions of frequency [2]. Also, sometimes, the desired transfer functions might be such that there is a finite gap between the pass band and stop band which may be called a "don't care" band where the transfer function can be arbitrary. Therefore, it is desirable to approximate an ideal filter by a continuous transfer function. A simple continuous approximation to the ideal low pass filter is the n^{th} order Butterworth filter defined by

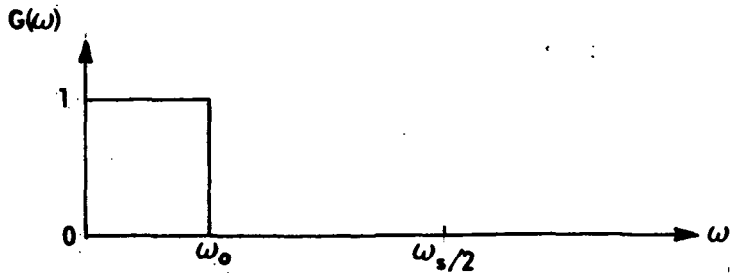
$$G(\omega) = \left[1 + (\omega/\omega_o)^{2n}\right]^{-\frac{1}{2}} \tag{3.1}$$

Where n is an integer and ω_o is the cut off frequency. The accuracy of the approximation to the ideal filter increases as n, the "order" of the filter increases.

Another type of approximation that can be used when there is a "don't care" band is [4]:

$$G(\omega) = \begin{cases} 0 & \text{for } |\omega| \geq \omega_T \\ 1 & \text{for } |\omega| \leq \omega_C \\ \left(\dfrac{\omega_T - |\omega|}{\omega_T - \omega_C}\right)^p & \text{for } \omega_C \leq |\omega| \leq \omega_T \end{cases} \tag{3.2}$$

where p is a positive integer and $[\omega_C, \omega_T]$ is the "don't care" band.



ω_s = SAMPLING FREQUENCY

FIGURE 3.1a IDEAL LOW PASS CHARACTERISTIC.

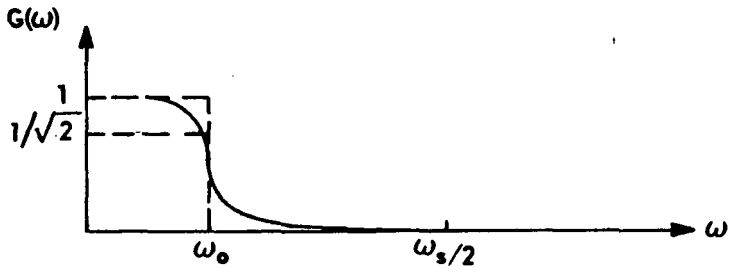


FIGURE 3.1b BUTTERWORTH APPROXIMATION.

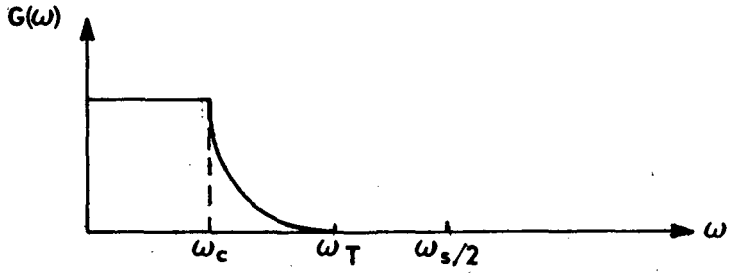


FIGURE 3.1c FREQUENCY RESPONSE DEFINED BY EQUATION 3.2

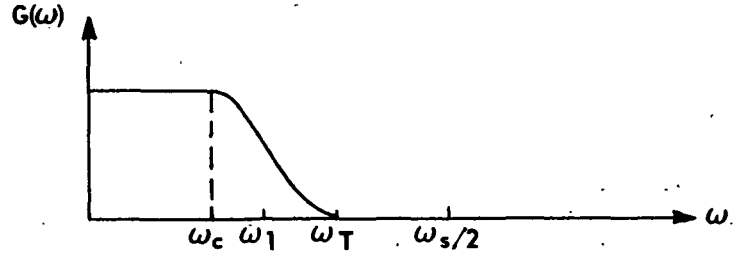


FIGURE 3.1d FREQUENCY RESPONSE DEFINED BY EQUATION 3.4

The function G (ω) given by (3.2) is continuous but does not have a continuous derivative. The following modification of G (ω) assures continuity of the first derivative for p $\geq$2:

$$G(\omega) = \begin{cases} 0 & \text{for } |\omega| \geq \omega_T \\ 1 & \text{for } |\omega| \leq \omega_C \\ \dfrac{(\omega_T - |\omega|)^p}{(\omega_T - \omega_C)\ (\omega_T - \omega_1)^{(p-1)}} & \text{for } \omega_1 \leq |\omega| \leq \omega_T \\ 1 - \dfrac{(|\omega| - \omega_C)^p}{(\omega_T - \omega_C)\ (\omega_1 - \omega_C)^{(p-1)}} & \text{for } \omega_C \leq |\omega| \leq \omega_1 \end{cases} \tag{3.3}$$

where ω_1 is an arbitrary number such that $\omega_C \leq \omega_1 \leq \omega_T$.

In particular, choosing $\omega_1 = (\omega_C + \omega_T)/2$ we get:

$$G(\omega) = \begin{cases} 0 & \text{for } |\omega| \geq \omega_T \\ 1 & \text{for } |\omega| \leq \omega_C \\ 2^{(p-1)} \left(\dfrac{\omega_T - |\omega|}{\omega_T - \omega_C} \right)^p & \text{for } \omega_1 \leq |\omega| < \omega_T \\ 1 - 2^{(p-1)} \left(\dfrac{|\omega| - \omega_C}{\omega_T - \omega_C} \right)^p & \text{for } \omega_C \leq |\omega| \leq \omega_1 \end{cases} \tag{3.4}$$

The forms of approximating characteristics represented by equations (3.1), (3.2) and (3.4) are shown in figures 3.1(b), (c) and (d) respectively.

Approximation to a high pass filter transfer function can be obtained by (1 - G(ω)) where G (ω) is an approximation to a low pass filter transfer function. Approximation to a band pass filter transfer function can be obtained as a difference between approximations to two low pass filter transfer functions.

Sometimes it might be desired to compensate for the errors induced by effects of the imaging system. If the transfer function is known, or may be estimated or measured and is represented by, say, G (ω), then, the required filter transfer function is H (ω) = 1/G (ω). For example, compensation for finite size of the scanning beam can be obtained using a filter transfer function shown in figure 3.2. [5]

Since various picture processing applications need several types of filter transfer functions it is desirable to have a general software package which can approximate any given filter transfer function. Such a package has been developed and is described in the following sections.

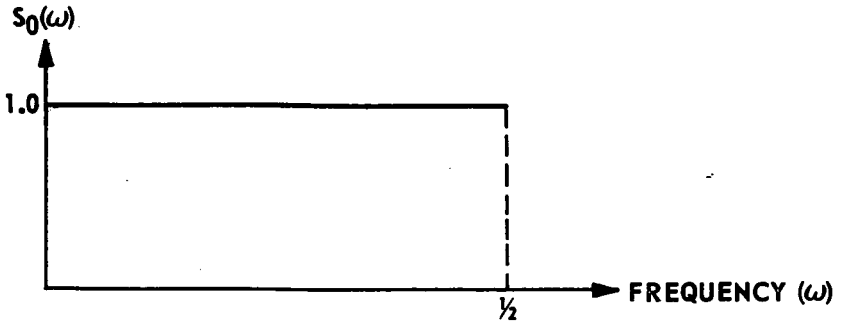


FIGURE 3.2a IDEAL SCANNING BEAM FREQUENCY RESPONSE.

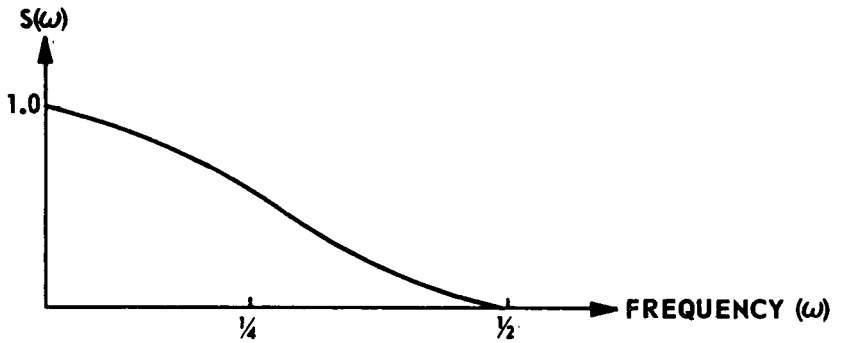
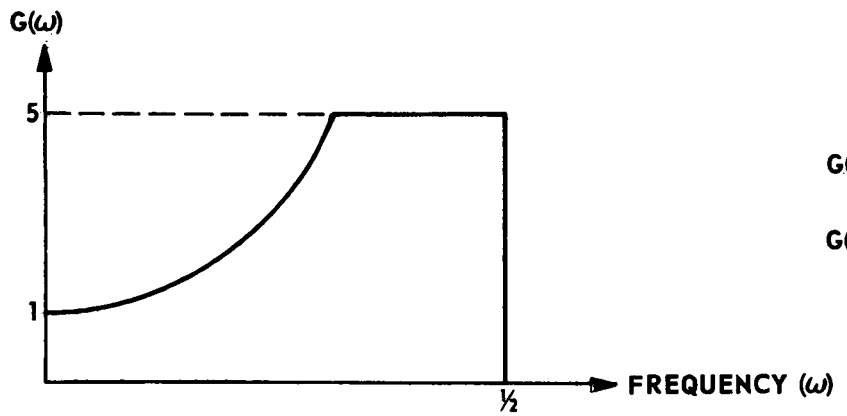


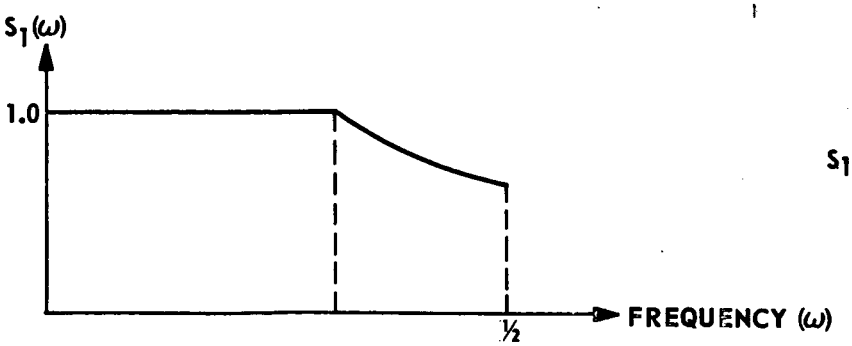
FIGURE 3.2b ACTUAL SCANNING BEAM FREQUENCY RESPONSE.



$G(\omega) = 1/S(\omega)$ WHENEVER $1/S(\omega) \leq 5$

$G(\omega) = 5$ WHEN $1/S(\omega) \geq 5$

FIGURE 3.2c COMPENSATING FILTER TRANSFER FUNCTION.



$S_1(\omega) = S(\omega)G(\omega)$

FIGURE 3.2d COMPENSATED SCANNING BEAM FREQUENCY RESPONSE.

First of all, the filter input/output relations are described for the continuous and discrete cases.

b. Filter equations. Consider first the equations for the continuous case. Let the intensity function input to the filter be given by $S(x)$. Let the filter transfer function be given by $G(\omega)$ and let $g(x)$ be the corresponding impulse response. Then, the output $S_o(x)$ from the filter is given by

$$S_o(x) = \int_{-\infty}^{\infty} g(u) \quad S(x-u)du \tag{3.5}$$

Now, if $\widetilde{S}(\omega)$ and $\widetilde{S}_o(\omega)$ are the Fourier transforms of the input and the output of the filter respectively, then it follows that

$$\widetilde{S}_o(\omega) = G(\omega) \quad \widetilde{S}(\omega) \tag{3.6}$$

The corresponding expressions for the discrete case are as follows:

$$S_{on} = \sum_{k=-\infty}^{\infty} g_k S_{n-k} \text{ for } -\infty < n < \infty \tag{3.7}$$

where $\{S_n\}$, $\{g_n\}$ and $\{S_{on}\}$ are samples of the input, filter impulse response and the output respectively at intervals of Δx.

If the input and the filter impulse response have spectra which are zero outside the interval $-1/(2\Delta x) \leq f \leq 1/(2\Delta x)$, then the expressions for the Fourier transforms have the form

$$\widetilde{S}(\omega) = \sum_{n=-\infty}^{\infty} S_n \exp(-j\omega n \Delta x) \tag{3.8}$$

$$G(\omega) = \sum_{n=-\infty}^{\infty} g_n \exp(-j\omega n \Delta x) \tag{3.9}$$

in the interval $-\pi/\Delta x \leq \omega \leq \pi/\Delta x$ and $\widetilde{S}(\omega) = G(\omega) = 0$ elsewhere. The output spectrum is also band limited and is given by

$$\widetilde{S}_o(\omega) = G(\omega)\widetilde{S}(\omega) \tag{3.10}$$

The numerical errors associated with assuming a function to be band limited and deriving equations (3.8), (3.9) and (3.10) are discussed by Cooley et al. [6]

The process of designing the digital filter consists in choosing the 'weights' g_n such that the desired band limited transfer function $G(\omega)$ is obtained. However, since the picture data to be filtered is a finite set of numbers and obviously an infinite summation cannot be computed on a computer, it is desirable to design filters with a finite set of weights g_k, $k = 0, \ldots, L-1$. Using this set of weights, the equation (3.7) can be written as

$$S_{on} = \sum_{k=0}^{L-1} g_k S_{n-k} \qquad n = 0, \ldots, N-1 \tag{3.11}$$

noting that the input and output are also finite sets of numbers.

Now, in order to obtain $G(\omega)$ exactly, in general, an infinite sequence of weights g_k is necessary. Hence, a finite sequence of weights will only provide an approximation to $G(\omega)$. Filters with smaller number of weights generally result in faster implementation but higher errors in approximation. There are several techniques in the literature for obtaining filter weights to approximate given transfer functions. Two of these methods will be described below. The first of these can be used to approximate an arbitrary continuous transfer function $G(\omega)$. Note that discontinuous transfer functions cannot be approximated to arbitrarily small errors even with an infinite sequence of filter weights [2].

c. Helm's 4-T's method [7] (transform-truncate-transform-test). Let $G(\omega)$ be the specified filter transfer function which is zero outside the interval

$$-\pi/\Delta x \leq \omega \leq \pi/\Delta x.$$

(a) First, a large number N of samples is chosen such that all the significant features of the frequency response such as the peaks, zeros and discontinuities are sufficiently accurately represented by the sampled response and a sequence $\{G_n\}$ is formed such that

$$G_n = G(\omega_n) \tag{3.12}$$

where

$$\omega_n = \frac{2\pi}{\Delta x}\,\frac{n}{N} \quad \text{for } n = 0, 1, \ldots, (N/2)-1$$

and

$$\omega_n = \frac{-2\pi}{\Delta x}\,\frac{N-n}{N} \quad \text{for } n = N/2, \ldots, N-1 \tag{3.13}$$

(b) Now, the IDFT (Inverse Discrete Fourier Transform) of the sequence $\{G_n\}$ $(n = 0,1,\ldots,N-1)$ is computed. The IDFT, denoted by $\{g'_n\}$ gives the impulse response corresponding to the frequency response $\{G_n\}$.

(c) A number $L<N$ is chosen. In the sequence $\{g'_n\}$, N-L contiguous values are set equal to zero, g_o' and g'_{N-1} being considered contiguous. The sequence $\{g_n\}$ is obtained by permuting $\{g'_n\}$ such that the values which were set to zero appear at the end. That is, $g_n = 0$ for $n=L,\ldots,N-1$.

(d) The DFT of the sequence $\{g_n\}$ is computed. Let this DFT be denoted by $\{G'_n\}$. The error e in the approximation is defined as a suitable function of $\{G_n - G'_n\}$. For example

$$e = \operatorname*{Max}_{0 \leq n \leq N-1} \left| G_n - G'_n \right| \tag{3.14}$$

(e) If the error e is not small enough the steps (c) and (d) are repeated with a larger value of L. If e is smaller than the specified error then L is decreased and the steps (c) and (d) are repeated. The new values of L can be obtained either by a successive bisection process or by a modified Newton-Raphson Method.

This method is illustrated in figure 3.3

d. Programs to implement the 4-T's method: A set of programs has been developed to implement the method described in the steps (a) through (e) in the previous subsection. The programs have two options. The first option is to double the value of L whenever the error e is greater than or equal to the specified error. When the value of L exceeds a prespecfied maximum value $\bar{L}$, L is taken equal to $\bar{L}$ and the impulse response is truncated to L terms and written as an output tape. The second option is to use successive bisection of intervals of L. First the error is found with a small initial value L_1 of L (which is less than $\bar{L}$). Next the error is formed for $L = \bar{L}$. Now for the i^{th} iteration the value of L_i is given by

$$L_i = (L_{i-1} + L_j)/2 \qquad \text{for } i \geq 3 \tag{3.15}$$

where

$$j = \text{Max} \left\{ j \,\middle|\, j < i-1 \text{ and } (e_{i-1} - E)\,(e_j - E) \leq 0 \right\} \tag{3.16}$$

and

e_i is the error during the i^{th} iteration and E is the given error.

Several types of filters have been designed using these programs. Figures 3.4, 3.5 and 3.6 show the magnitude versus frequency plots of some examples of the original transfer functions for low, high and band pass filters respectively and the approximating transfer functions produced by the above method. The magnitude characteristics for these filters were assumed to be derived by a basic low pass characteristic given by equation (3.2). The maximum permissible error e in the equation (3.14) was assumed to be 0.05 (i.e. maximum gain of approximately -26 db in the stop band and an error of 0.22 db in the pass band). The number N of samples of the frequency response was taken to be 256. The parameters defining the filters shown in figures 3.4, 3.5, and 3.6 and the values of error for the attempted values of L are shown in tables 3.1, 3.2 and 3.3 respectively. The first option (doubling L until error is below the specified value) was used in these cases.

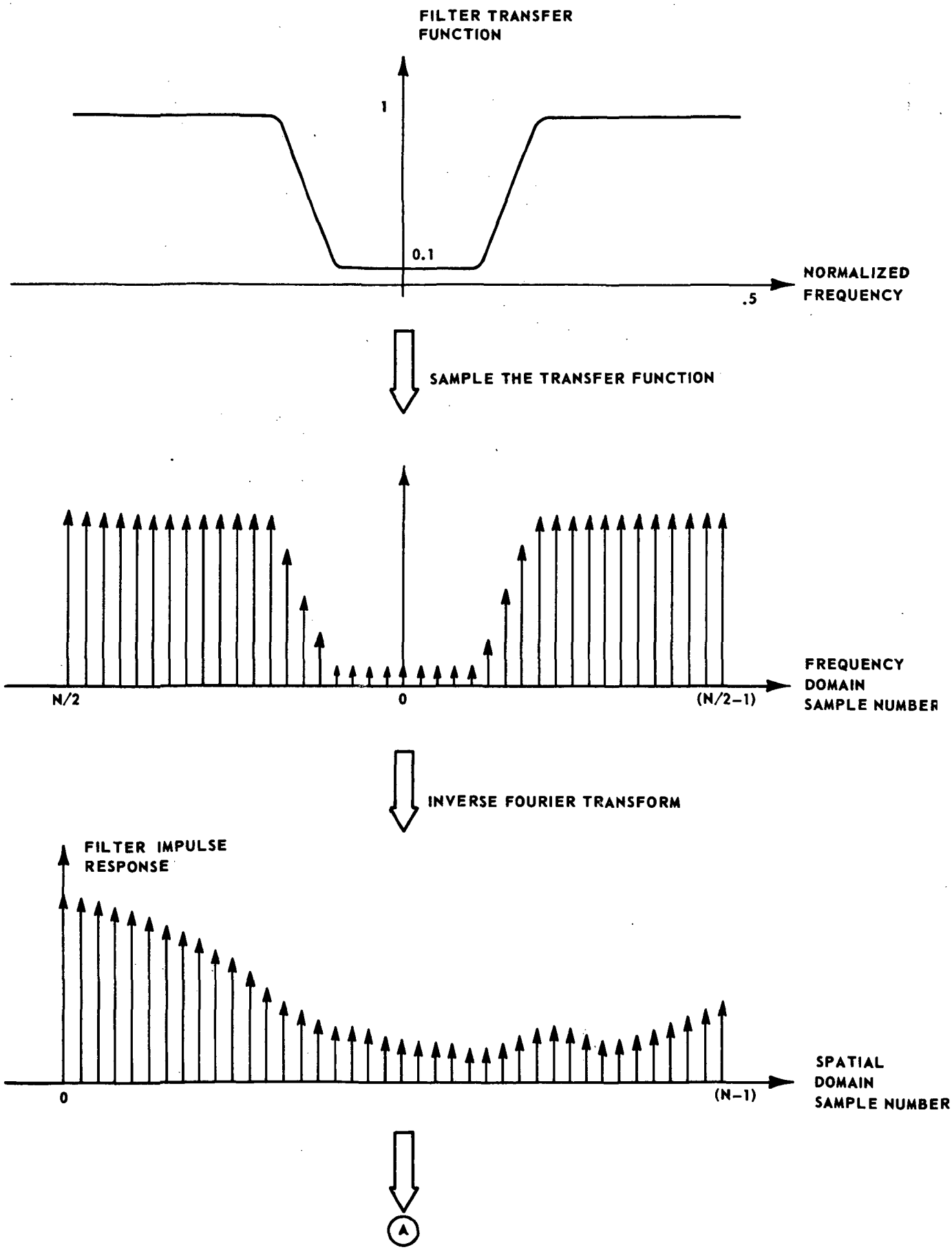


FIGURE 3.3 STEPS IN FILTER DESIGN (4-T METHOD)

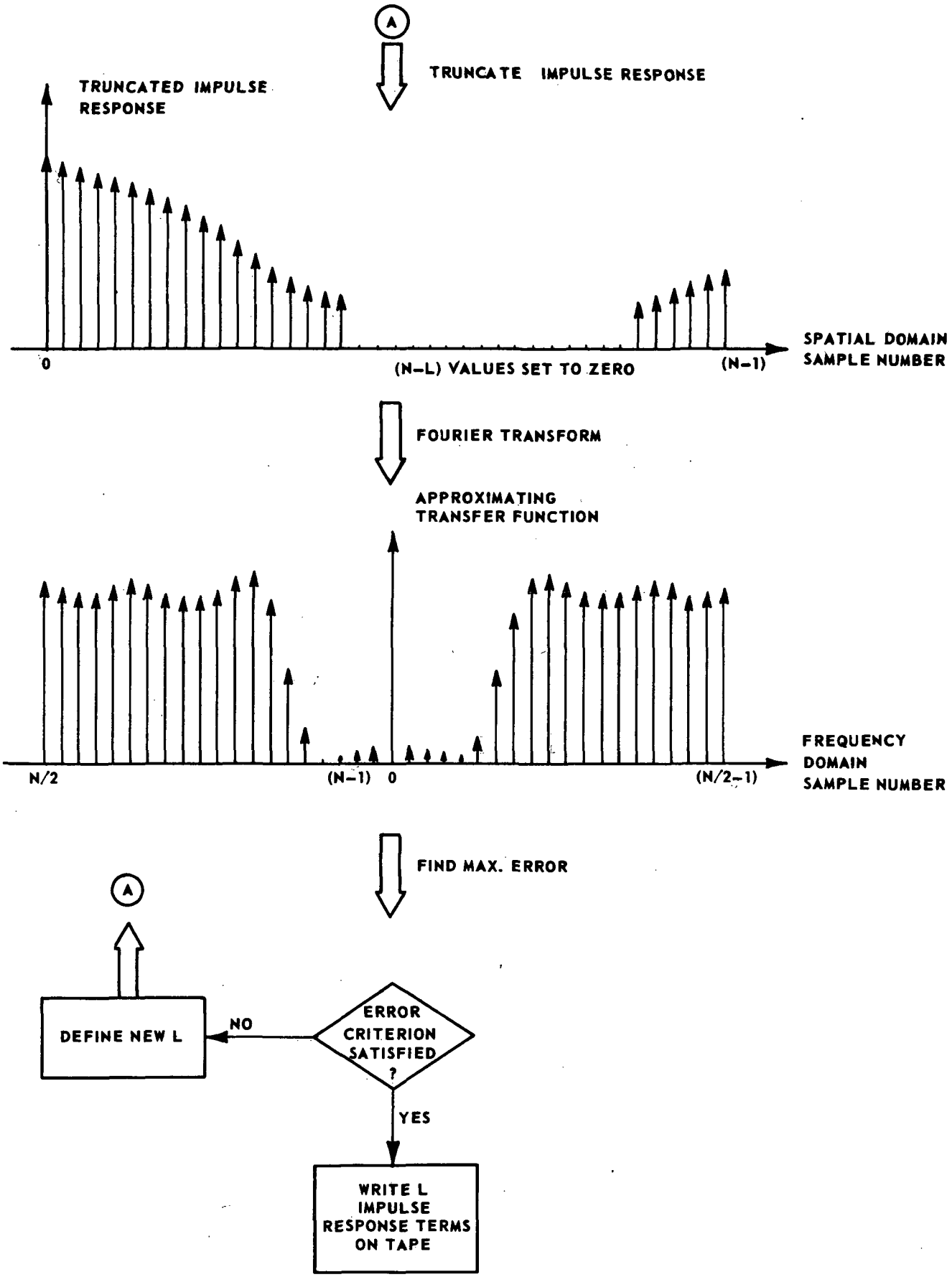


FIGURE 3.3 (CONTINUED) STEPS IN FILTER DESIGN (4–T METHOD)

TABLE 3.1 APPROXIMATIONS TO A LOW PASS FILTER

Pass Band = $[0, 0.15\omega_s]$, Stop Band = $[0.25, 0.50\omega_s]$ (The pass band and stop band are joined by a third order polynomial as in equation (3.2))

Filter Length L	Error e
8	0.3265
16	0.1633
32	0.0728
64	0.0285

TABLE 3.2 APPROXIMATIONS TO A HIGH PASS FILTER

Pass Band = $[0.25\omega_s, 0.50\omega_s]$, Stop Band = $[0, 0.15\omega_s]$ (The pass band and stop band are joined by a third order polynomial as in equation (3.2))

Filter Length L	Error e
8	0.3265
16	0.1633
32	0.0728
64	0.0285

TABLE 3.3 APPROXIMATIONS TO A BAND PASS FILTER

Pass Band = $[0.10\omega_s, 0.25\omega_s]$, Stop Band = $[0, 0.50\omega_s] \cup [0.30\omega_s, 0.5\omega_s]$ (The first and second stop bands are connected to the pass band by a 2nd and 3rd order polynomial respectively.

Filter Length L	Error e
8	0.3766
16	0.3000
32	0.1820
64	0.0895
128	0.0371

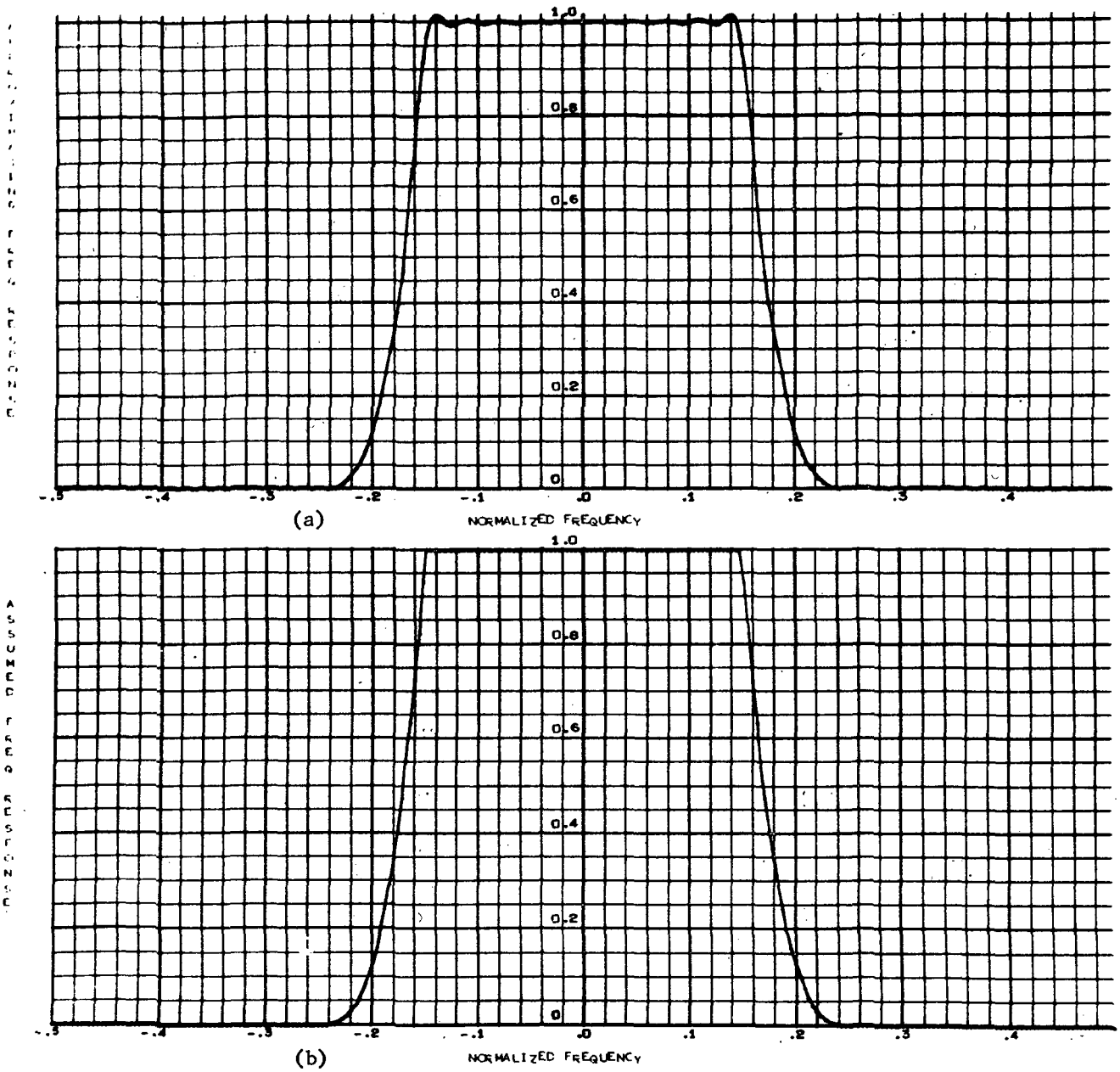


Figure 3.4 Low Pass Filter

(a) Approximating Frequency Response

(b) Specified Frequency Response

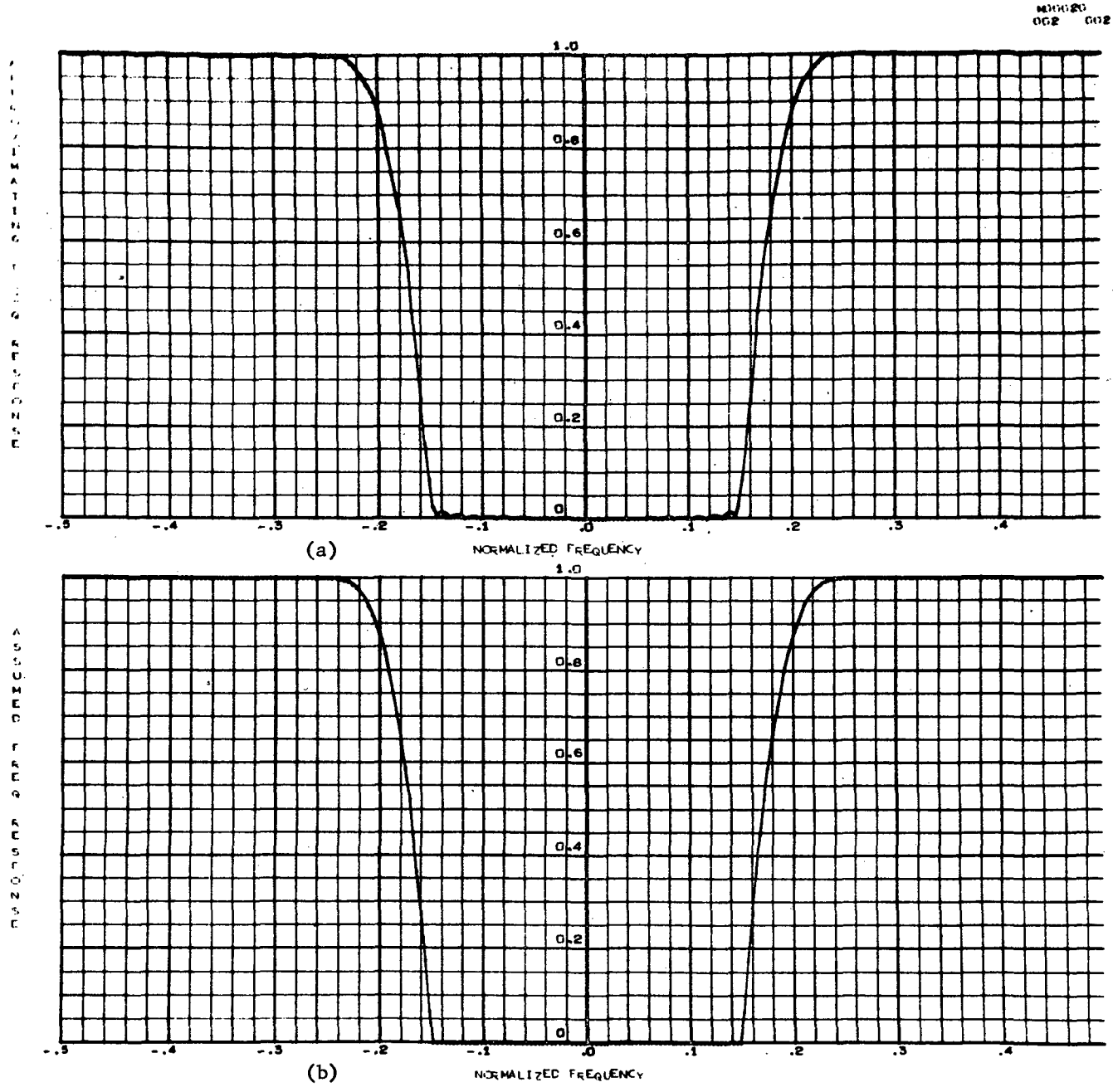


Figure 3.5 High Pass Filter

(a) Approximating Frequency Response

(b) Specified Frequency Response

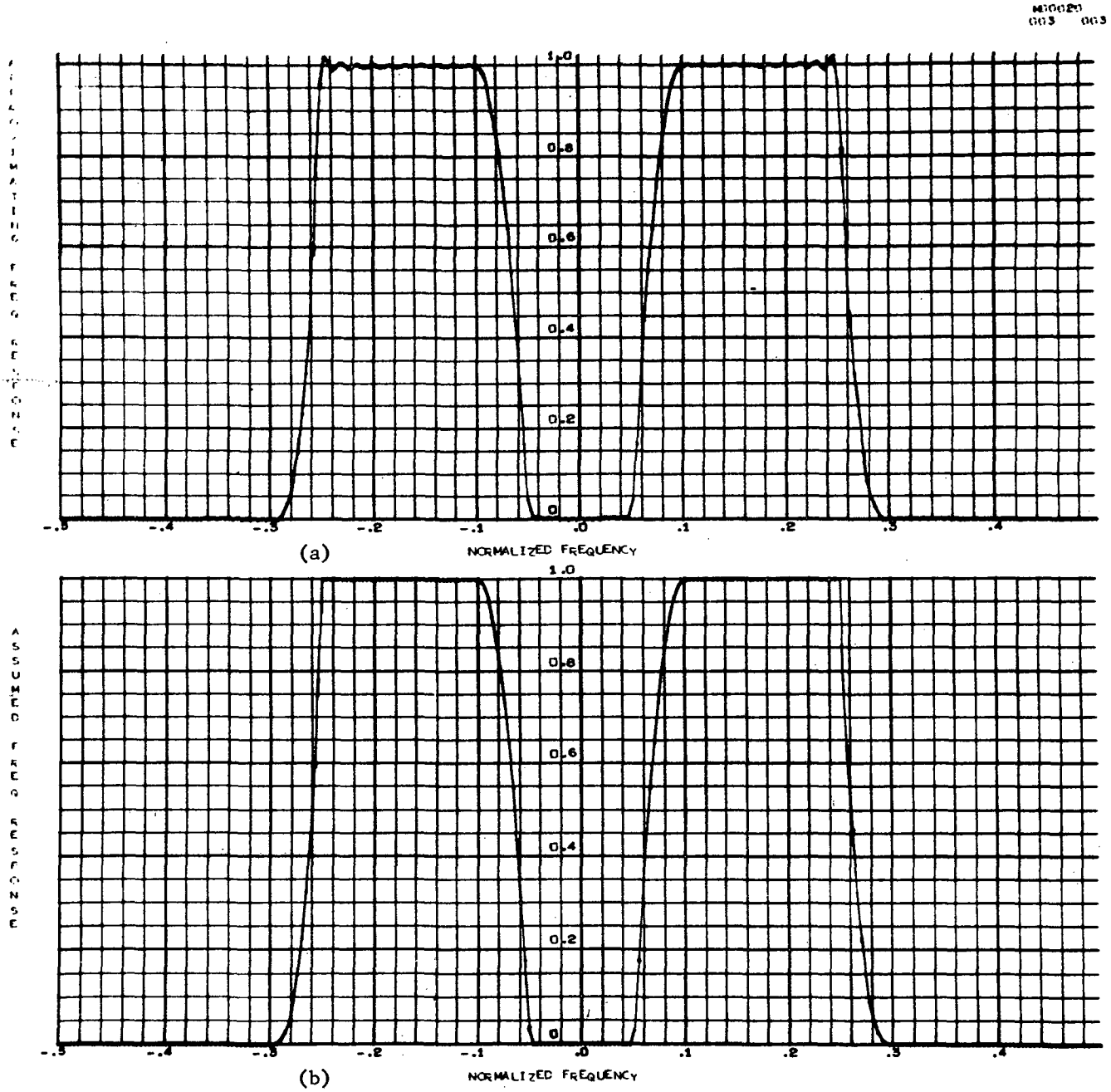


Figure 3.6 Band Pass Filter

(a) Approximating Frequency Response

(b) Specified Frequency Response

e. Frequency sampling method of design [8]: Helm's 4-T's method described above first finds the filter impulse response and truncates it to a desired degree of approximation in order to obtain a short filter. Another technique for approximating a class of filter transfer functions is due to Rabiner et al [8] who uses a frequency sampling approach. In this method the resulting impulse response is 'folded' or 'aliased' rather than being truncated. The steps in the design procedure are summarized below.

(a) A set of frequencies is chosen at which the sampled frequency response is specified. This set consists of a number of points much smaller than that required for an accurate representation of the desired frequency response. The values of the transfer function at some of the frequencies in the above set are left as design parameters. For example, in the design of an ideal low pass filter the transfer function at the pass band frequencies is fixed at 1 and at the stop band frequencies is fixed at 0, but a few frequencies are considered transition frequencies joining the pass and stop bands and the values of the transfer function at these frequencies are considered design parameters.

(b) The values of the continuous frequency response are obtained in terms of assumed values of the design parameters by using either an explicit formula (given by a sampling theorem) or interpolation using fast Fourier transform or chirp z-transform algorithm. Here, continuous frequency response means the response at a number of frequencies which is much larger than the number of frequency samples chosen in step (a) and adequate to characterize the necessary features of the desired transfer function.

(c) Using the interpolated frequency response the values of the design parameters are readjusted until an error criterion is satisfied. The error criterion, for example, could be that the maximum of the absolute value of the difference between the approximating and the desired frequency responses in a given frequency range be a minimum or in the case of an ideal low pass filter that the maximum gain in the stop band should be a minimum.

(d) When the error criterion has been satisfied, the final values of the free parameters are used along with the fixed values of sampled transfer function which were determined in step (a).

Rabiner et al have used this method to design a large number of low pass and band pass filters and wide band differentiators. They have provided tables in reference [8] giving design data. These tables give the band width, total number of frequency samples used, the values of transition samples and the minimum gain in the stop band achieved for the low pass and band pass filters. High pass filter frequency samples can be obtained from these tables by subtracting all the low pass filter samples from 1. The frequency samples for band widths not listed in the tables can be obtained by linear interpolation of the tabulated data. It has been found experimentally [8] that the deviation of the maximum gain in the stop band obtained by linear interpolation will be less than 6 db from the optimum.

2. Spatial Domain Approach: The main difference between the frequency domain approach to filter design and the spatial domain approach occurs in the way the desired characteristics of the filter are specified. In cases where the signal and noise spectra are known and are approximately non-overlapping it is simple to choose a filter transfer function. This would be a frequency domain approach. On the other hand, in some cases such as matched filter design it is much simpler to specify the filters directly in terms of the weights in the spatial domain. Also, nonlinear and spatially variant filtering operations cannot be specified conveniently in the frequency domain. This subsection presents two types of filters which are designed using the spatial domain approach.

a. Matched filters (template matching): Matched filters have traditionally been used for detection of signals in presence of noise in communication systems [9-11]. These are derived in communication theory for the one dimensional case and are optimal in the sense that they maximize the signal to noise ratio. A derivation of the matched filter for the two dimensional case which is useful for picture processing is given below for the sake of completeness. (See reference [12] for a more detailed discussion). The continuous case is considered first and the corresponding equations for the discrete case follow easily.

Let $f_I(x,y)$ be the input signal and let the corrupting noise be $n(x,y)$. Assume that $n(x,y)$ is additive noise generated by a wide sense stationary random process which is ergodic in its autocorrelation. Let $R_N(\tau,T)$ be the auto correlation and $S_{NI}(u,v)$ be the power spectrum of noise. Let $h(x,y)$ be the impulse response of the filter.

Now, the input to the filter is given by

$$f(x,y) = f_I(x,y) + n(x,y) \tag{3.17}$$

The output of the filter due to the signal is given by $f_o(x,y) = f_I(x,y) * h(x,y)$ where * denotes convolution. Therefore, the output signal has instantaneous energy at the point (ξ,η) given by

$$\begin{aligned} S &= \left| f_o(\xi,\eta) \right|^2 \\ &= \left| \int\int F_o(u,v) \exp\{ j(u\xi + v\eta) \} \, du \, dv \right|^2 \\ &= \left| \int\int F_I(u,v) H(u,v) \exp\{ j(u\xi + v\eta) \} \, du \, dv \right|^2 \end{aligned} \tag{3.18}$$

where F_I, F_o and H are the Fourier transforms of f_I, f_o and h respectively. The power spectrum of the output of the filter due to noise input is given by

$$S_{NO}(u,v) = S_{NI}(u,v) \left| H(u,v) \right|^2 \tag{3.19}$$

Therefore, the signal to noise ratio of the output is given by

$$\frac{S}{N} = \frac{\left| \int\int F_I(u,v)\, H(u,v)\, \exp\{j(u\xi + v\eta)\}\, du\, dv \right|^2}{\left| \int\int S_{NI}(u,v) \left| H(u,v) \right|^2 du\, dv \right|} \tag{3.20}$$

Now, it is well known (see [3], for example) that the power spectrum $S_{NI}(u,v)$ is non-negative for all u,v. In particular, assume that $S_{NI}(u,v) > 0$ for all u,v. (This is certainly true for the case of white noise input. For a general treatment of the case where $S_{NI}(u,v)$ is permitted to be zero for some values of u,v see [13]). Then, using the Cauchy-Schwarz inequality on the numerator of the right side of equation (3.20), it follows that

$$\frac{S}{N} \leq \frac{\int\int \left| F_I(u,v) \right|^2 / S_{NI}(u,v)\, du dv \int\int S_{NI}(u,v) \left| H(u,v) \right|^2 du dv}{\int\int S_{NI}(u,v) \left| H(u,v) \right|^2 du dv} \tag{3.21}$$

That is,

$$\frac{S}{N} \leq \int\int \left| F_I(u,v) \right|^2 / S_{NI}(u,v)\, du\, dv \tag{3.22}$$

From (3.20) and (3.22) it is easy to show that the maximum value of the output signal to noise ratio is achieved if and only if

$$H(u,v) = \lambda F_I^*(u,v) \exp(-j(u\xi + v\eta)) / S_{NI}(u,v) \tag{3.23}$$

which is the condition for (3.22) to hold with equality, where λ is a constant with respect to u and v and the asterisk denotes complex conjugation. If the noise is assumed to be white, then the expression (3.23) reduces to

$$H(u,v) = \lambda F_I^*(u,v) \exp(-j(u\xi + v\eta)) \tag{3.24}$$

and the impulse response (i.e. point spread function) of the filter can be written as

$$h(x,y) = \lambda f_I(\xi - x, \eta - y) \tag{3.25}$$

Therefore, for the particular case of additive white noise in the input, the output signal to noise ratio at the point (ξ,η) is maximized if the filter point spread function "matches" the input signal shifted to the point (ξ,η) and reflected point by point in (ξ,η) to within a multiplicative constant. In other words, if the location or locations of a particular image pattern is to be detected in the presence of additive white noise one would just move a template having that pattern over all points of the given picture and find the locations at which the maxima of the cross correlation between the given image and the template occur. Further, if contrast variations within a picture need to be compensated for, then the maxima of <u>normalized</u> cross correlation should be examined.

The discrete version of the matched filter equations can be written down immediately. We shall assume here that only the variations of the picture function over the local average are significant, so that the template can be assumed to have zero mean. A set of weights $\{g_{k\ell}\}$ are chosen with $(k, \ell) \in [-K,K] \times [-L,L]$ so that they represent the shape of the pattern to be detected. The input picture (i.e., the picture to be filtered) is given by $\{x_{ij}\} = \{s_{ij} + n_{ij}\}$ where n_{ij} is white noise which is ergodic in its mean and variance and statistically independent of s_{ij}. Also, n_{ij} has zero mean. The normalized cross correlation between $\{g\}$ and $\{x\}$ is given by

$$y_{ij} = \sum_{k,\ell} g_{k\ell} x_{i+k, j+\ell} / (\|g\| \, \|x_{ij}\|) \tag{3.26}$$

where

$$\|g\| = (\sum_{k,\ell} g_{k\ell}^2)^{\frac{1}{2}} \tag{3.27a}$$

and

$$\|x_{ij}\| = (\sum_{k,\ell} x_{i+k, j+\ell}^2)^{\frac{1}{2}} \tag{3.27b}$$

Now, the expected value of y_{ij} is maximum when $s_{i+k, j+\ell} = \lambda g_{k\ell}$ for $k = -K, \ldots, K$ and $\ell = -L, \ldots, L$. Further, due to the assumptions on the noise, the actual values of y_{ij} will be approximately equal to the expected value for sufficiently large values of K and L and is given by

$$y_{ij} \approx \frac{\sum_{k,\ell} g_{k\ell} s_{i+k, j+\ell}}{\|g\| (\|s_{ij}\|^2 + \|n_{ij}\|^2)^{\frac{1}{2}}} \tag{3.28}$$

When a 'match' occurs, the value of y_{ij} is given by

$$y_{ij} \approx \frac{\|g\|}{(\|g\|^2 + \|n\|^2)^{\frac{1}{2}}} \tag{3.29}$$

noting that $\|n_{ij}\|$ is approximately constant with respect to i and j and denoting it by $\|n\|$. Thus, if one knows the statistics of noise, one can use equation (3.29) to obtain a threshold on the normalized cross correlation to decide whether the template matches at a particular location.

In recent years, the application of matched filtering for optical character recognition and image pattern recognition has become very common [1, 12, 14, 15]. Unlike other image enhancement techniques matched filtering does not, in general, produce an output image which looks like the pattern being enhanced. Instead, it produces a "correlogram" from which only the locations of template match can be determined. One can then reconstruct the desired image using the correlogram and the template. In some cases, however, the correlogram itself (appropriately thresholded) is a reasonable representation of the desired image. A particular case where this is true is when lines of a specified width in a particular direction are to be enhanced.

b. Density Manipulations: Even though all filtering operations described so far are in fact manipulations of the densities in the input picture to get an output picture, we shall specialize the term density manipulations to mean point operations (in contrast with local operations) where the output due to a particular picture point depends on the density at the point only (and not on the densities at some of the neighboring points also). Such operations are sometimes useful for image enhancement and are much faster than local operations.

(1) Table look up method: When the input data consists of only a small set of integer values compared to the total number of data points the density transformation can be implemented by defining it as a table. When the input data is read the output is found by merely looking up the table. This is a very fast method since no arithmetic operations are involved. Examples of this method are (i) contrast stretching by subtracting the minimum density number from the density number at a point and linearly rescaling the density numbers over the available range (0 to 63) (ii) modifying the distribution of densities by remapping of densities, say, to yield a linear distribution curve.

(2) Linear and non-linear scaling: More often than not, when image data is filtered, the resulting set of numbers are both positive and negative and are in floating point format (particularly in high pass and band pass filtering). Therefore, in order to be able to display the filtered image it is necessary to convert the numbers to integers between 0 and 63. In order to obtain as much contrast as possible in the displayed image we set the smallest number in the array to be 0 and the largest to be 63 and rescale the data using a monotonic and non-decreasing function. The monotonic function could be either linear or nonlinear (square law, logarithmic, exponential, etc.) depending on whether contrast stretch should be uniform over the density range or whether contrasts in some density ranges are to be emphasised to a greater extent than in others. Sometimes it might be desirable to set all data below a certain value to 63 and scale the data in between according to a monotonic scaling function.

A program "SCALE" has been designed to take all these possibilities into account. This program handles the input-output operations, computes output using an externally specified scaling function, rounds off the computed values to integers, truncates the resulting set of integers (if necessary) to lie between 0 and 63 and, as an option, computes, the maximum, minimum and mean of the input file which can be used in the scaling function. This program can handle several input files on tape at a time without having to rewind the tape several times. (The input tapes will be rewound only once if the option to compute maxima, minima and means is used).

An example of a function which can be used as an external function in the routine SCALE is given below:

$$y_{ij} = \alpha_1 + \alpha_2 \{(x_{ij}-\alpha_3) / (\alpha_4-\alpha_3)\}^{p} \qquad (3.30)$$

where α_1 is a bias parameter, $\alpha_2 > 0$ is the contrast factor, α_3 is the background to be subtracted and α_4 can be taken as

$$\alpha_4 = \text{Max} \left(\left| \text{Max } x_{ij} \right| , \left| \text{Min } x_{ij} \right| \right) \tag{3.31}$$

the maximum and minimum being over ij taking on all values within the domain of data.

B. Filter Implementation Techniques

A general approach to linear shift invariant filter implementation would be to store the two dimensional filter (either the impulse response or the frequency response) in core and find the convolution of the data with the filter by bringing into core the appropriate sections of data. This is the approach taken in most installations where special purpose hardware is available for computing convolutions and disk files are available for intermediate storage and retrieval of data. However, on a tape based system it is simpler to do filtering in one dimension so that the data from tape may be read, filtered and written on output tape sequentially, one record (i.e. one picture line) at a time. Two dimensional filtering may then be performed as a two pass procedure. This restricts the class of filters which can be implemented to 'separable' filters whose two dimensional transfer function $H(u,v)$ can be written as a product $H_1(u)H_2(v)$ or as a sum $H_1(u) + H_2(v)$. The two pass procedures for the product and sum type separable filters are shown in figures 3.7 and 3.8 respectively.

There are several methods of implementation of one dimensional digital filters. These can be broadly classified as nonrecursive and recursive implementations. Both the implementations are discussed below.

1. Nonrecursive Implementations: The output of a one dimensional linear filter is expressed in nonrecursive realizations by the formula

$$y_i = \sum_{k=0}^{L-1} g_k \, x_{i-k} \tag{3.32}$$

The values of y_i in (3.32) can be computed directly by multiplying the terms of $\{g\}$ and $\{x\}$ term by term and adding them. This would involve L multiplications and (L-1) additions for every point of output. The number of arithmetic operations can be reduced considerably in the case of a filter whose length is much shorter than the length of an input data record by using the Fast Fourier Transform (FFT) algorithm. A method which uses the FFT is due to Stockham [16] and Helms [17] and is called the "select-saving" method. The steps in this method are summarized below:

(a) A number L' is chosen where $L' \geq L$. The choice of an optimum L' is governed by the radix of the FFT algorithm which determines the total number of arithmetic operations required.

(b) The sequence $\{g\}$ is appended with L'-L zeros at the end. Denote the new sequence by $\{g'\}$. Also let the first L' terms of the sequence

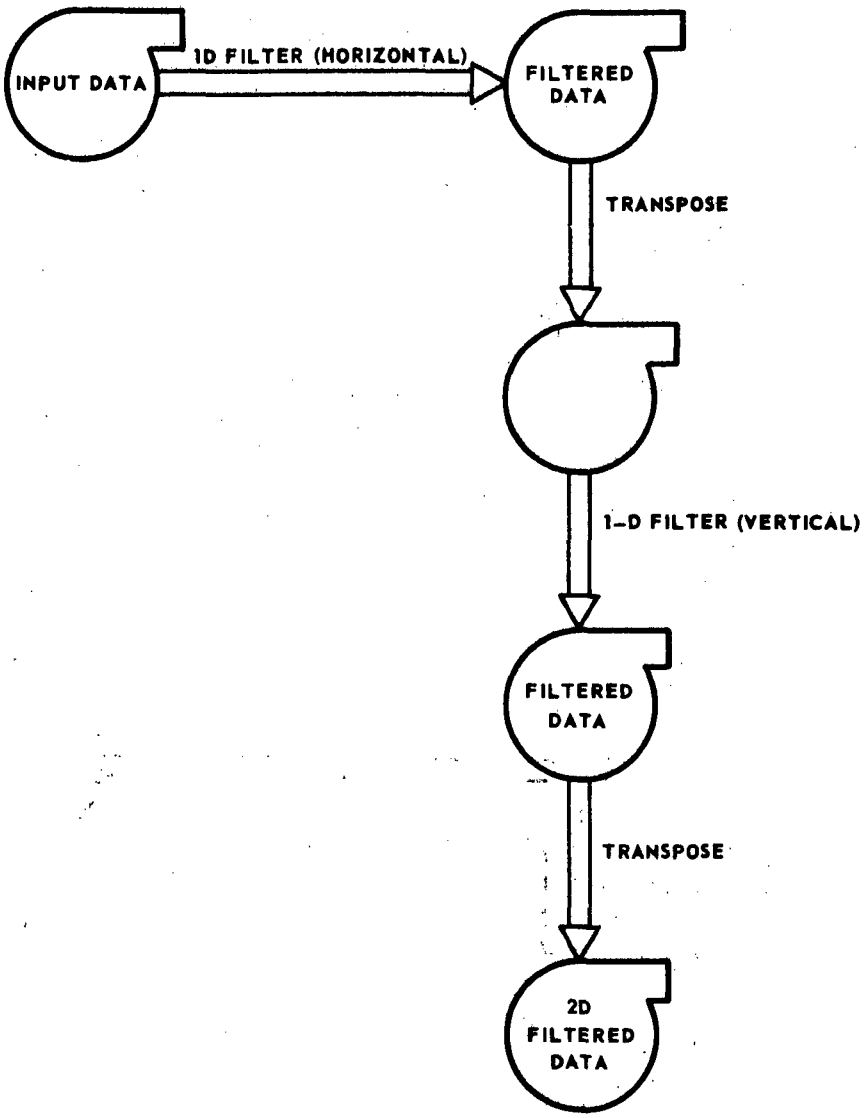


FIGURE 3.7 TWO PASS PROCEDURE FOR PRODUCT TYPE FILTERS

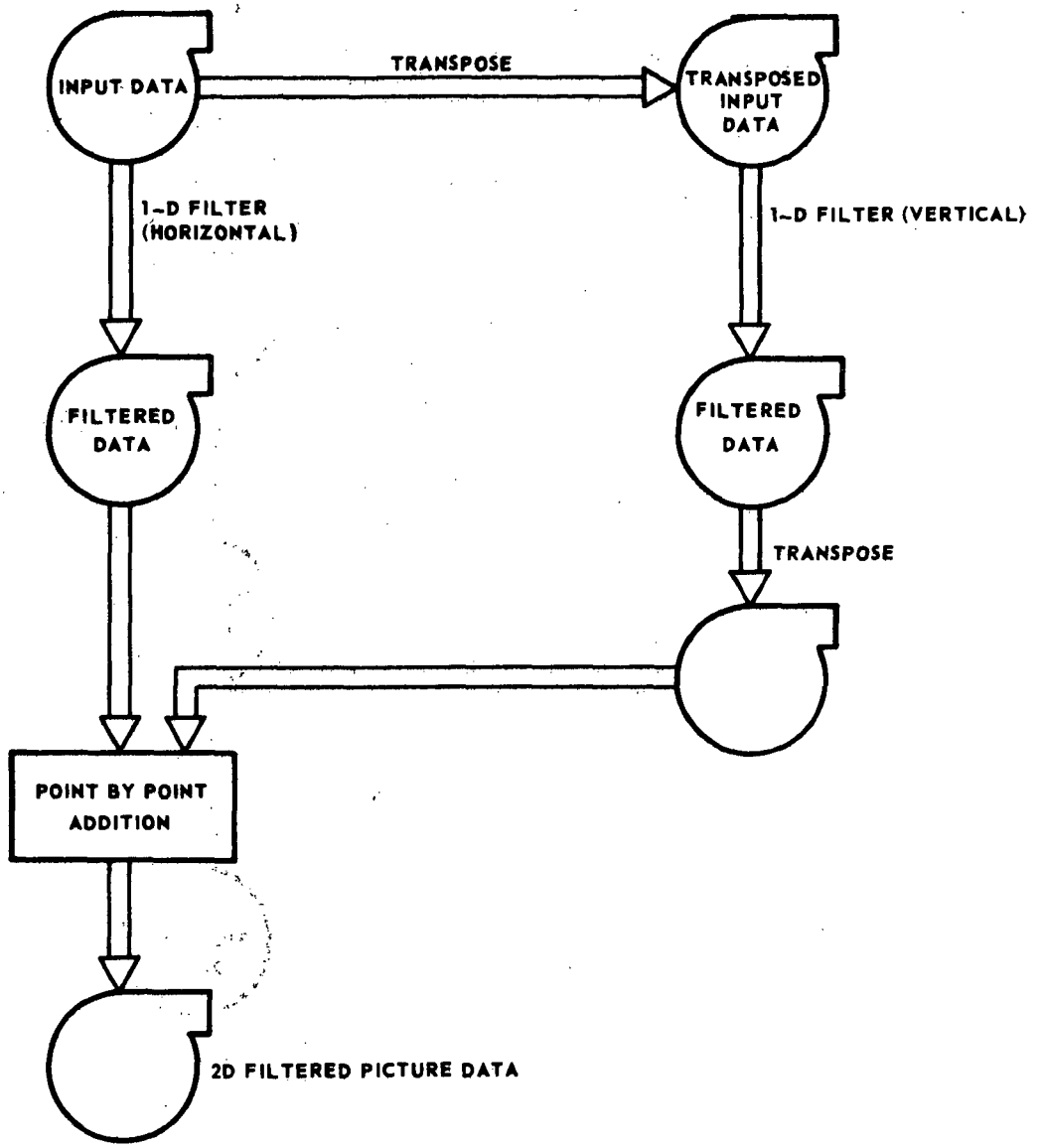


FIGURE 3.8 TWO PASS PROCEDURE FOR SUM TYPE FILTERS

$\{x\}$ be denoted by $\{\bar{x}\}$.

(c) The Discrete Fourier Transforms (DFT's) of $\{g'\}$ and $\{\bar{x}\}$ are computed using FFT.

(d) The IDFT $\{z\}$ of the product of the DFT's above is computed.

(e) It can be shown that due to the cyclic nature of the DFT only the values z_m for m = L-1, L,..., L'-1 are the valid values of convolution. Therefore only those values are retained.

(f) Now, $\{\bar{x}\}$ is redefined by discarding its first L'-L+1 values, reducing the subscripts of the remaining L-1 values by L'-L+1 and appending the next L'-L+1 values of the sequence $\{x\}$ to be original L-1 values of $\{\bar{x}\}$.

(g) The steps (a) through (f) are repeated until all the values of $\{x\}$ have been used.

The number of arithmetic operations required for computing the convolution for N points is approximately 3K additions and 2K multiplications where

$$K = 2NL'(\log_2 L')/(L'-L+1) \tag{3.33}$$

This shows for long filters it is advantageous to use the select-saving method and the advantage over direct convolutions increases as the length of the filter increases. Now, the number of output values obtained with the select-saving method is equal to (N-L+1) when filtering a record of length N with a filter with L weights. The output obtained is given by the equations

$$\begin{aligned} y_0 &= x_0 g_{L-1} + x_1 g_{L-2} + \cdots + x_{L-1} g_0 \\ &\vdots \\ y_{N-L} &= x_{N-L} g_{L-1} + x_{N-L+1} g_{L-2} + \cdots + x_{N-1} g_0 \end{aligned} \tag{3.34}$$

However, it is desirable to obtain an output array consisting of the same number of values as the input array so that the filtered picture and the original picture have the same size. To do this, it is necessary to append the input array suitably with (L-1) extra components (which may be chosen either zero or some other numbers, essentially to make the filtered picture look reasonable even at the edges). Also it is convenient to store the filter weights generated by the 4-T's method (Section III A.1.c) approximating the desired transfer function of the filter to within a known linear phase difference, i.e., to within a known shift in the spatial domain. In order to take these factors into account, two new arrays x'_n and y'_n are defined as shown below and select-saving method is used with x' and g as input and y' as output.

Define the sequence $h = h_0, \ldots, h_{N_1-1}$ as the set of weights giving the desired approximations to the filter transfer function found by Helm's 4-T's method, where $h_i = \ldots = h_{(i+N_1-L-1)} = 0$. Then, these weights are stored by writing i, L and the L "consecutive" non-zero weights $\{ g \}$ given by

$$
\begin{aligned}
g_0 &= h_{i+N_1-L} \\
g_1 &= h_{i+N_1-L+1} \\
&\cdot \\
&\cdot \\
&\cdot \\
g_{L-i-1} &= h_{N_1-1} \\
g_{L-i} &= h_0 \\
g_{L-i+1} &= h_1 \\
&\cdot \\
&\cdot \\
&\cdot \\
g_{L-1} &= h_{i-1}
\end{aligned}
\tag{3.35}
$$

The output will be free from shift if

$$
\begin{aligned}
y_0 &= \ldots + h_i x_{-1} + h_0 x_0 + h_{N_1-1} x_1 + \ldots \\
&\cdot \\
&\cdot \\
&\cdot \\
y_k &= \ldots + h_1 x_{k-1} + h_0 x_k + h_{N_1-1} x_{k+1} + \ldots \\
&\cdot \\
&\cdot \\
&\cdot \\
y_{N-1} &= \ldots + h_1 x_{N-2} + h_0 x_{N-1} + h_{N_1-1} x_N + \ldots
\end{aligned}
\tag{3.36}
$$

where x_k are to be suitably defined for $k \notin [0, N-1]$.

Now, the filter output due to input $\{ x'_n \}$ using select-saving method is given by

$$
y'_k = x'_k g_{L-1} + x'_{k+1} g_{L-2} + \ldots + x'_{k+L-1} g_0 \tag{3.37}
$$

for $k = 0, 1, \ldots, N-1$.

Comparing (3.35), (3.36) and (3.37), it follows that the output in (3.36) is obtained if

$$
\left.\begin{aligned}
x'_0 &= x_{-(i-1)} \\
&\vdots \\
x'_{i-2} &= x_{-1}
\end{aligned}\right\} \text{ To be defined}
$$

$$
\begin{aligned}
x'_{i-1} &= x_0 \\
x'_i &= x_1 \\
&\vdots \\
x'_{i+N-2} &= x_{N-1}
\end{aligned}
\qquad (3.38)
$$

$$
\left.\begin{aligned}
x'_{i+N-1} &= x_N \\
&\vdots \\
x'_{N+L-2} &= x_{(N+L-2)-(i-1)}
\end{aligned}\right\} \text{ To be defined}
$$

and

$$y_k = y'_k \quad \text{for } k = 0, 1, \ldots, N-1$$

The program which implements the above nonrecursive filtering technique provides for three types of definition for x_k when $k \notin [0, N-1]$:

(i) periodic repitition with period N, i.e., $x_{-1} = x_{N-1}$, $x_N = x_0$, etc.

(ii) reflection about end points, i.e., $x_{-1} = x_1$, $x_N = x_{N-2}$, etc.

(iii) zero outside the given range, i.e., $x_{-1} = 0$, $x_N = 0$, etc.

This filter implementation was tested by filtering a square wave record consisting of 256 points using a high pass filter. The input record consisted of four consecutive values of 22 followed by four values of 18 and so on. The high pass filter used is shown in figure 3.5. The original data and the filtered data are shown in figures 3.9, 3.10, and 3.11 respectively for the

three methods of appending the edges mentioned above.

The DFT (256 point) of the original data shown in Figures 3.9, 3.10 and 3.11 is given by

$$X_0 = 5120;\ X_{32} = \frac{256}{1-\exp(j\pi/4)} = X^*_{224} \qquad X_{96} = \frac{256}{1-\exp(j3\pi/4)} = X^*_{160}$$

and

$$X_n = 0 \quad \text{for all other} \quad n \in [0,\ 255].$$

Now, the values of the filter transfer function corresponding to nonzero values of the input spectrum are given by (see Figure 3.5).

$$G_0 = 0;\ G_{32} = G^*_{224} = 0\ ;\ G_{96} = G^*_{160} = 1$$

Therefore, the DFT of the output is given by

$$Y_{96} = Y^*_{160} = \frac{256}{1 - \exp(j3\pi/4)}$$

and $Y_n = 0$ for all other $n \in [0,\ 255]$.

Taking the IDFT of $\{Y_n\}$, it is seen that

$$y_0 = 1;\ y_1 = -\frac{1}{1+\sqrt{2}};\ y_2 = -\frac{1}{1+\sqrt{2}};\ y_3 = 1$$

The values of y_n derived above are correct for all $n \in [0,\ 255]$ under the assumption that the sequence $\{x_n\}$ is periodic with period 256. This can be seen in Figure 3.9 which corresponds to making the data periodic with period 256.

However, in Figures 3.10, and 3.11, the values of y_n derived above are valid only in the middle regions where the 'edge' effects are absent. Note that the edge effect is more pronounced in Figure 3.11 than in Figure 3.10. This happens because introducing zeros at the ends of the data sequence causes large jumps at the ends while reflecting the data about the ends does not produce such severe jumps.

This nonrecursive filter implementation was also used on a test pattern made up of density variations along a record according to square waves at various frequencies. The pictures of the test pattern before and after filtering are shown in figures 3.12, 3.13, 3.14 and 3.15. The low, high and band pass filters used in these pictures are the ones shown in figures 3.4, 3.5 and 3.6 respectively. Note that the filtering is only one dimensional. The low pass filter has a smoothing effect which worsens the contrast of the original picture and the high pass filter produces contrast enhancement. The band pass filter enhances the contrast, but not as effectively as the high pass filter, since the highest of the frequencies which contribute to the sharpness of the edges have been filtered out. Note that in the high and band pass filtered pictures long horizontal edges

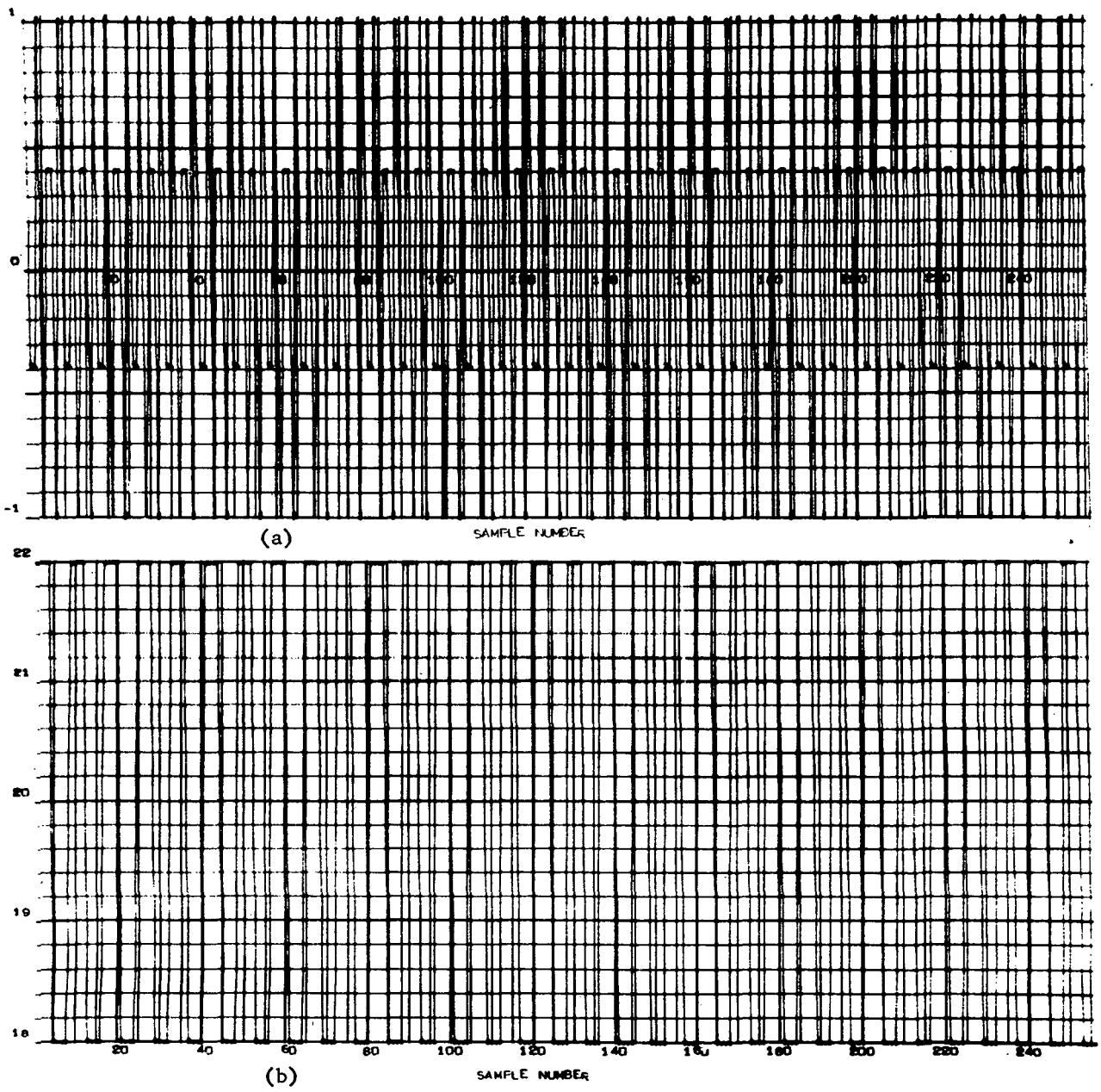


Figure 3.9 High Pass Filtering of a Test Pattern
(edges appended by periodic repetition)

(a) Filtered Signal (b) Original Signal

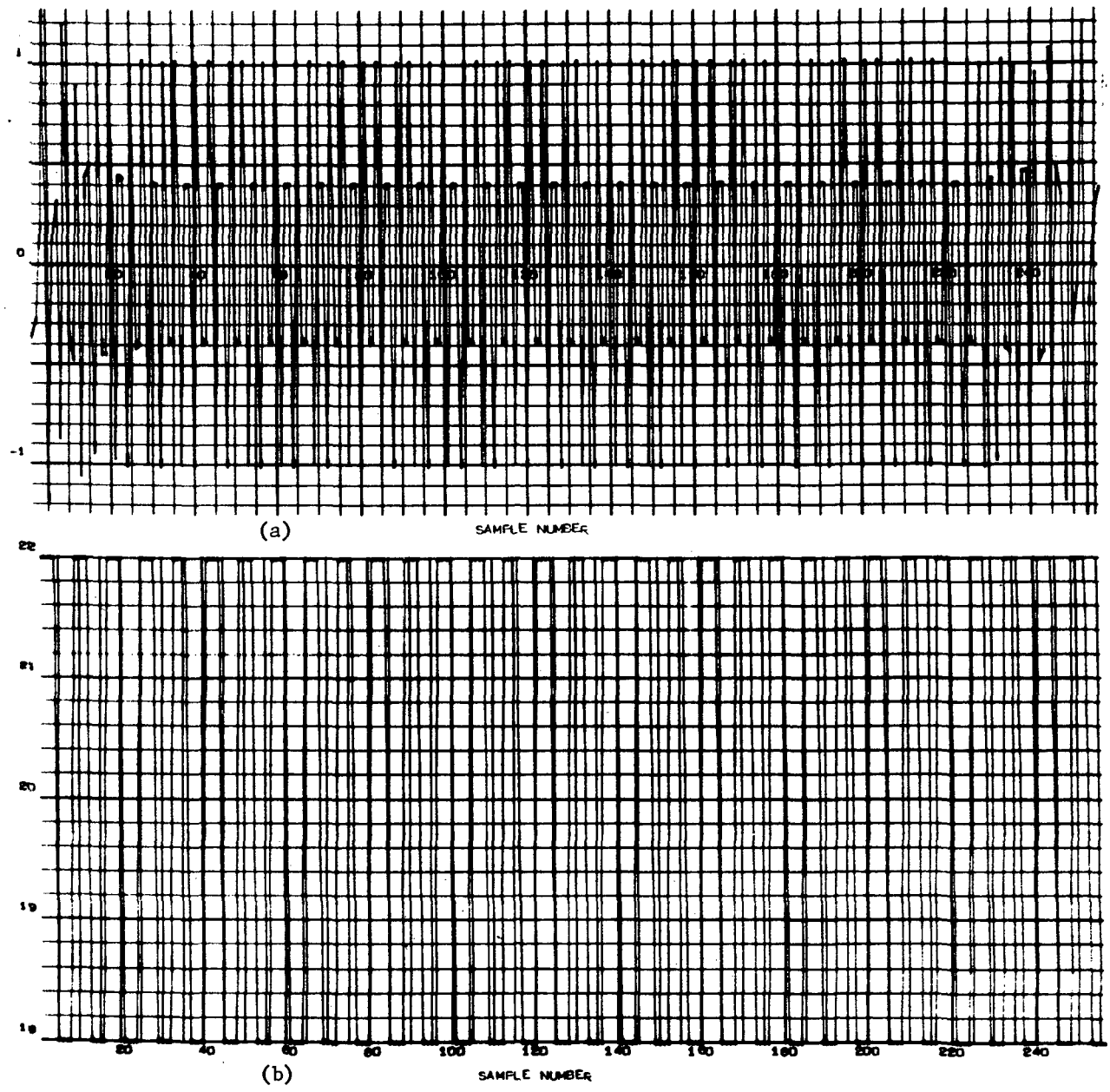


Figure 3,10 High Pass Filtering of a Test Pattern
(edges appended by reflection)

(a) Filtered Signal (b) Original Signal

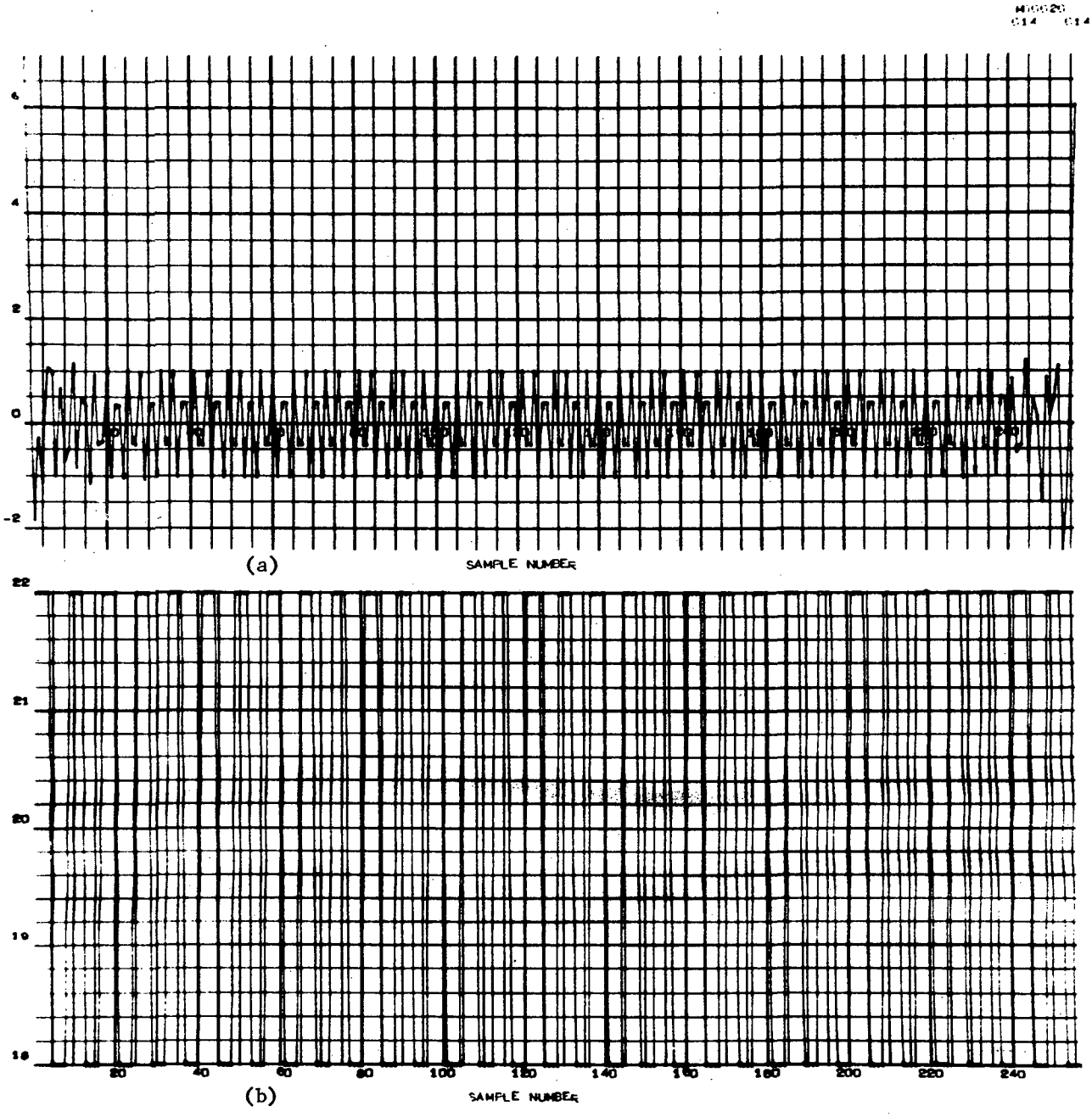


Figure 3.11 High Pass Filtering of a Test Pattern
(edges appended with zeros)

(a) Filtered Signal (b) Original Signal NASA—MSFC

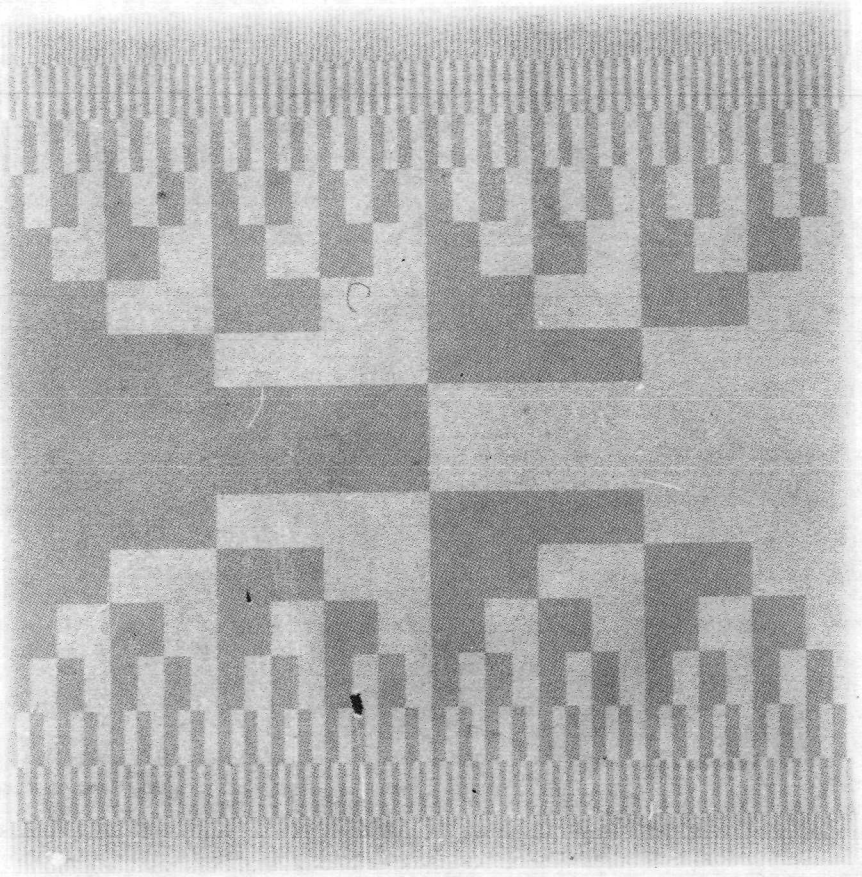


Figure 3.12 A Test Pattern Generated with Square Waves

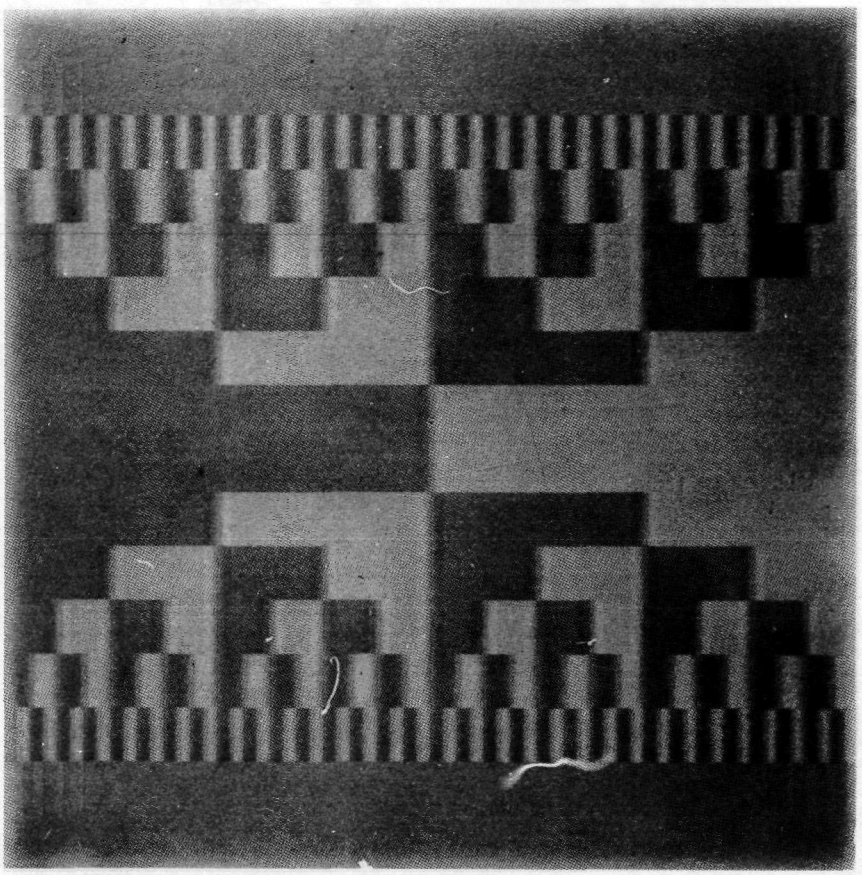


Figure 3.13 Low Pass Filtered Test Pattern

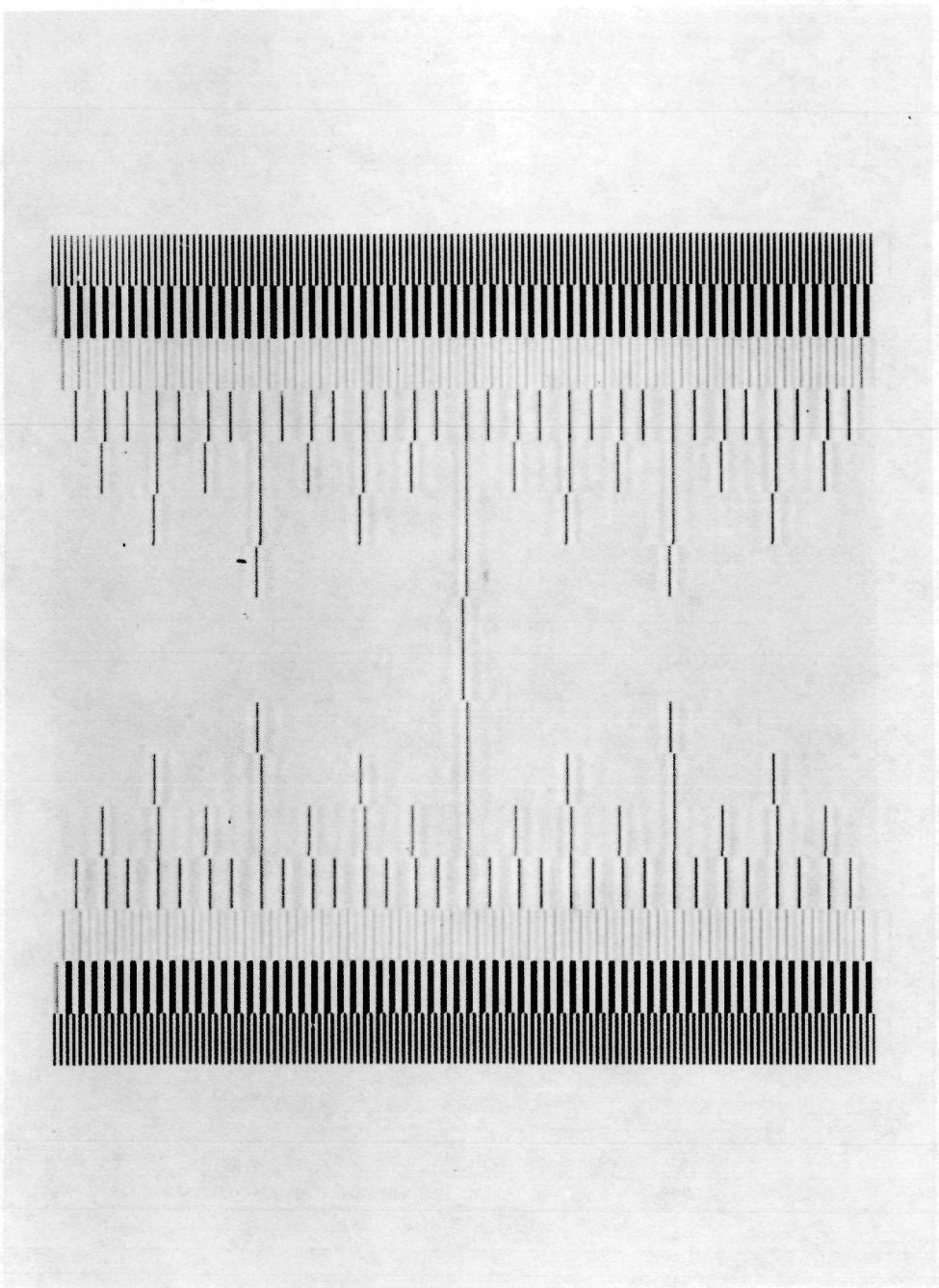


Figure 3.14 High Pass Filtered Test Pattern

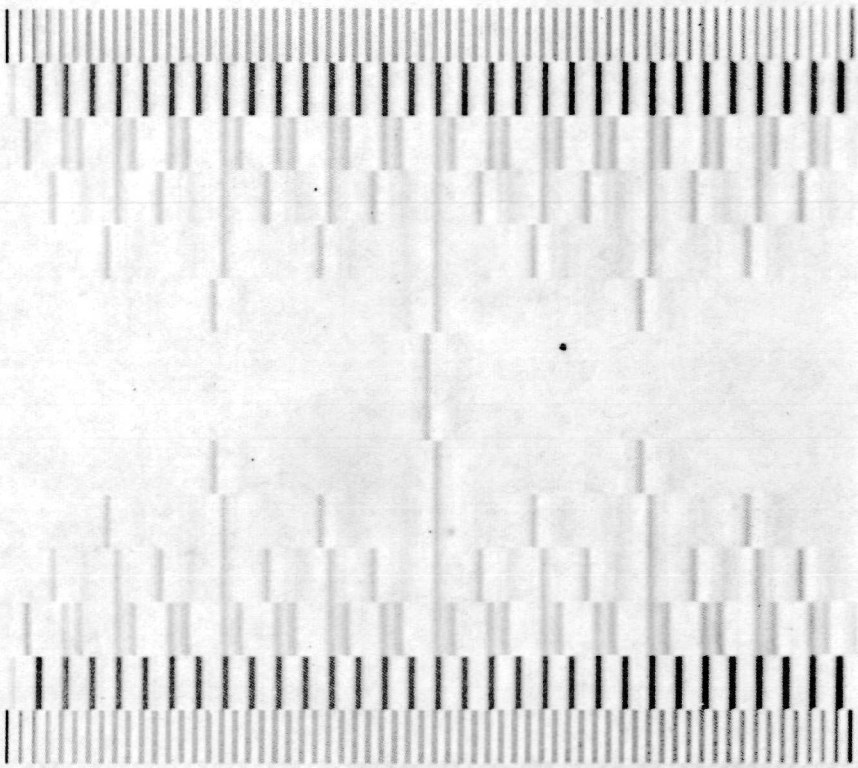


Figure 3.15 Band Pass Filtered Test Pattern

are missing since they are of zero frequency in the direction in which filtering is carried out.

Also note the absence of the vertical lines corresponding to the highest frequency in the low and band pass filtered pictures (the top and bottom $1/16^{ths}$ of the pictures).

Figure 3.16 shows the effect of band emphasis filtering on the test pattern of Figure 3.12. The filter transfer function used is the same as in Figure 3.6 except that the stop band gain is changed to 0.12 from 0. In this figure the vertical edges are enhanced and the horizontal edges are also preserved.

2. Recursive Implementations: The basic difference between nonrecursive and recursive implementations is that in recursive implementations the output computed at any stage is dependent on at least one of the previously computed values of the output. That is, the equation giving the output is of the form

$$\sum_{j=0}^{M-1} h_j \, y_{i-j} = \sum_{k=0}^{L-1} g_k \, x_{i-k} \tag{3.38}$$

where $h_0 = 1$ (with no loss of generality) and there exists at least one j such that $1 \leq j \leq M-1$ and $h_j = 0$.

It has been shown by Rabiner and Schafer [18] that narrow band filters designed by the frequency sampling techniques (section III.A.1.e) can be implemented recursively faster than the fast convolution method.

In this section we describe another class of filters which lend themselves to very fast recursive implementations even in two dimensions and also show how they can be used to generate matched filters.

a. Spatial and frequency domain characterization of the filters: First of all, consider the one-dimensional filter which replaces every pixel density by the average of the densities of $2L + 1$ neighboring pixels including the pixel itself. That is, if x_i is the input sequence and y_i is the output sequence, then

$$y_i = \left(\sum_{k=-L}^{L} x_{i+k} \right) / (2L + 1) \tag{3.39}$$

Thus, the filter weights are g_i where

$$\begin{aligned} g_i &= 1/(2L + 1) && \text{for} \quad i \in [-L, L] \\ &= 0 && \text{for} \quad i \notin [-L, L] \end{aligned} \tag{3.40}$$

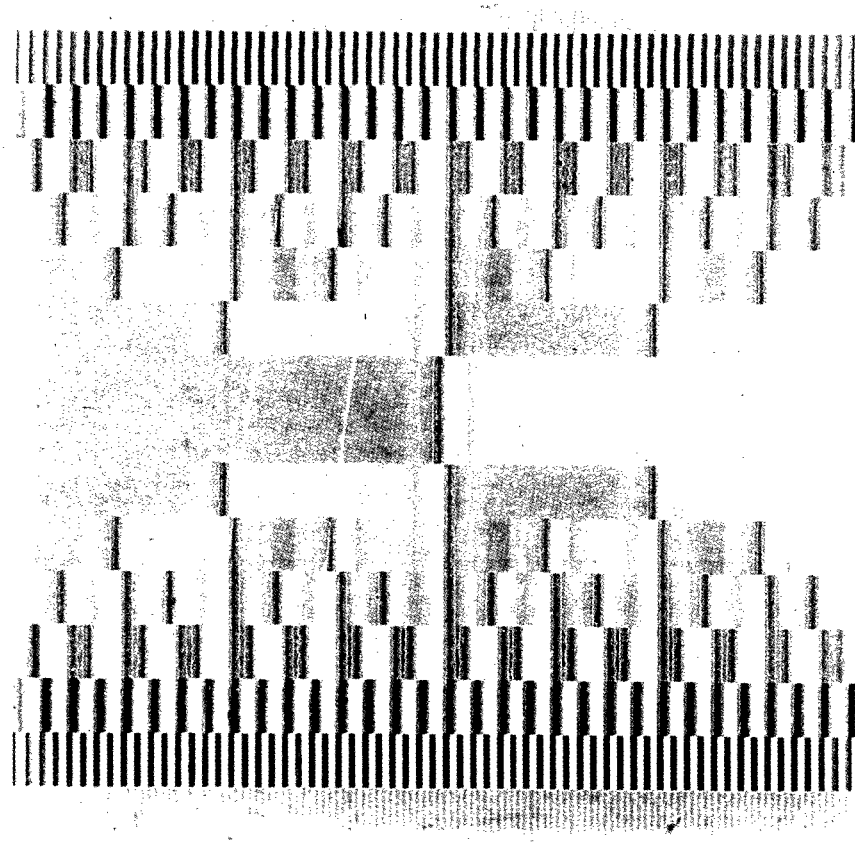


Figure 3.16 Band Emphasis Filtered Test Pattern

This filter is a linear phase filter whose modulation transfer function is shown in Figure 3.17 for L=3. It can be seen from Figure 3.17 that the 'major lobe' is at the low frequency end and the first zero is at 1/(2L+1) times the sampling frequency. Thus the filter is, in effect, a low pass filter, even though it is not a good approximation to an ideal low pass filter.

Now, from the above low pass filter it is easy to obtain a high pass filter by subtracting the output of the low pass filter from the input. That is, the average density of (2L + 1) neighboring pixels is subtracted from each pixel density. If h_i are the weights of this filter, we can write

$$\begin{aligned} h_i &= \delta_i - 1/(2L+1) \quad \text{for } i \in [-L, L] \\ &= 0 \quad \text{for } i \notin [-L, L] \end{aligned} \tag{3.41}$$

where

$$\delta_i = 1 \text{ for } i = 0 \text{ and } \delta_i = 0 \text{ for } i \neq 0.$$

The modulation transfer function of this filter is shown in Figure 3.18 for L = 20.

In general, the filter transfer function is derived from the filter weights by employing the discrete Fourier transforms such that the (complex) transfer function is

$$G(\omega_k) = \sum_{i=-L}^{L} g_i \exp(-ji\Delta k\ \omega_0) \tag{3.42}$$

where Δ is the sampling interval in the original domain and ω_0 is the sampling interval in the transform (frequency) domain. The zero frequency behavior of the filter is given by $G(\omega_k)$ evaluated for k = 0. Ideally, a low pass filter has unit transmission at zero frequency while the high pass filter has zero transmission at this frequency. Thus general constraints on the filter weights are

$$\sum_{i=-L}^{L} g_i = 1 \quad \text{for a low pass filter} \tag{3.43}$$

$$\sum_{i=-L}^{L} g_i = 0 \quad \text{for a high pass filter} \tag{3.44}$$

and these constraints are seen to be satisfied by the filter weights of Equations (3.40) and (3.41).

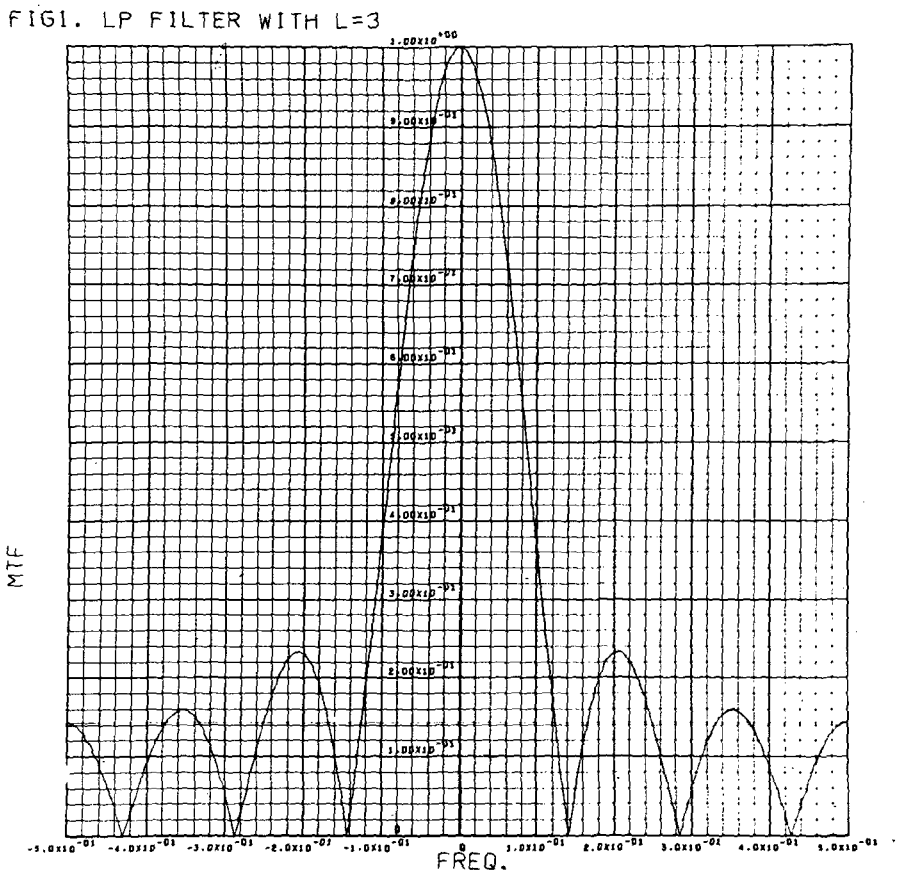


Figure 3.17 Low Pass Filter with L=3

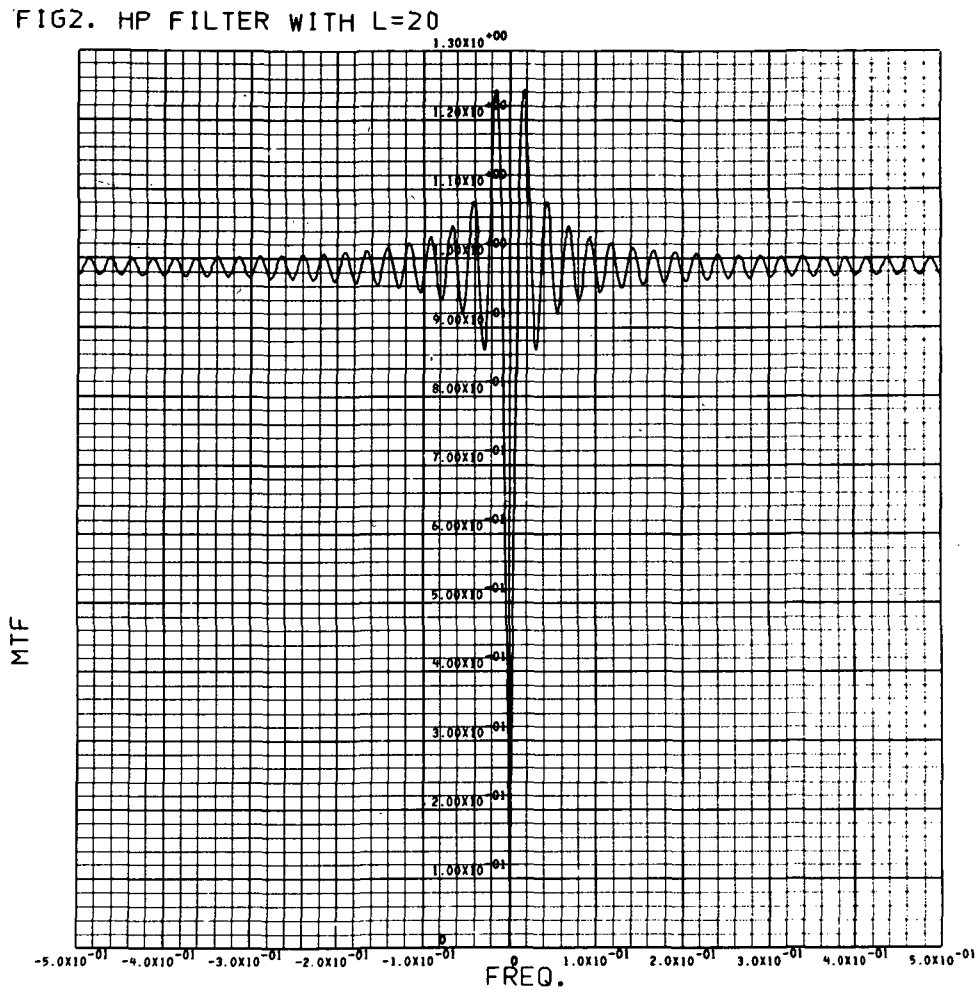


Figure 3.18 High Pass Filter with L=20

In discussing the design of the class of filters based upon constant weight low pass and high pass elements, it is convenient to represent the filter function graphically in terms of the signs of the weights. This representation is shown in Figure 3.19 for the basic low pass and high pass filter functions, and it will be employed extensively in the remainder of this discussion. Filter sample values may be assumed to be zero if no sign is associated with the sample. It should be obvious from Figure 3.19 that similarity of sign does not imply equality of weight, but the value of weight to be associated with a given sign can generally be evaluated simply.

The basic low pass filter of Equation (3.40) can be used to produce a band pass filter using two approaches. One approach is to have two low pass filters g_i' and g_i'' as in Equation (3.40) of lengths $(2K + 1)$ and $(2L + 1)$ respectively where $K < L$ and to subtract the output of the second from that of the first. This, in effect, produces a filter whose weights are

$$\begin{aligned} h_i &= \frac{1}{2K+1} - \frac{1}{2L+1} && \text{for } i \in [-K, K] \\ &= -\frac{1}{2L+1} && \text{for } i \in [-L, L] - [-K, K] \\ &= 0 && \text{for } i \notin [-L, L] \end{aligned} \tag{3.45}$$

The modulation transfer function corresponding to the weights in Equation (3.45) is shown in Figure 3.20 for $K = 3$ and $L = 20$.

Another approach for generating a band pass filter is to use a low pass filter (or Equation (3.40)) and a high pass filter (of Equation (3.41)) in tandem. Then, the modulation transfer function of the band pass filter will be the product of those of the low and high pass filters. The filter weights in this case are obtained by

$$h_i = \sum_{k=-\infty}^{\infty} g_k h'_{i-k} \qquad \text{for all } i \tag{3.46}$$

where $\{g_i\}$ are the low pass filter weights given by (3.40) with K instead of L and $\{h_i\}$ are the high pass filter weights of Equation (3.41) and $K < L$. It is easy to see that

$$\begin{aligned} h_i &= \frac{1}{2K+1} - \frac{1}{2L+1} && \text{for } i \in [-K, K] \\ &= -\frac{1}{2L+1} && \text{for } i \in [-K-1, -L+K] \cup [K+1, L-K] \\ &= \frac{1}{2L+1}\left(1 - \frac{|i| - (L-K)}{2K+1}\right) && \text{for } i \in [-L-K, -L+K-1] \cup [L-K+1, L+K] \\ &= 0 && \text{for } i \notin [-L-K, L+K] \end{aligned} \tag{3.47}$$

The modulation transfer function of this type of band pass filter is shown in Figure 3.21 for $K = 3$ and $L = 20$.

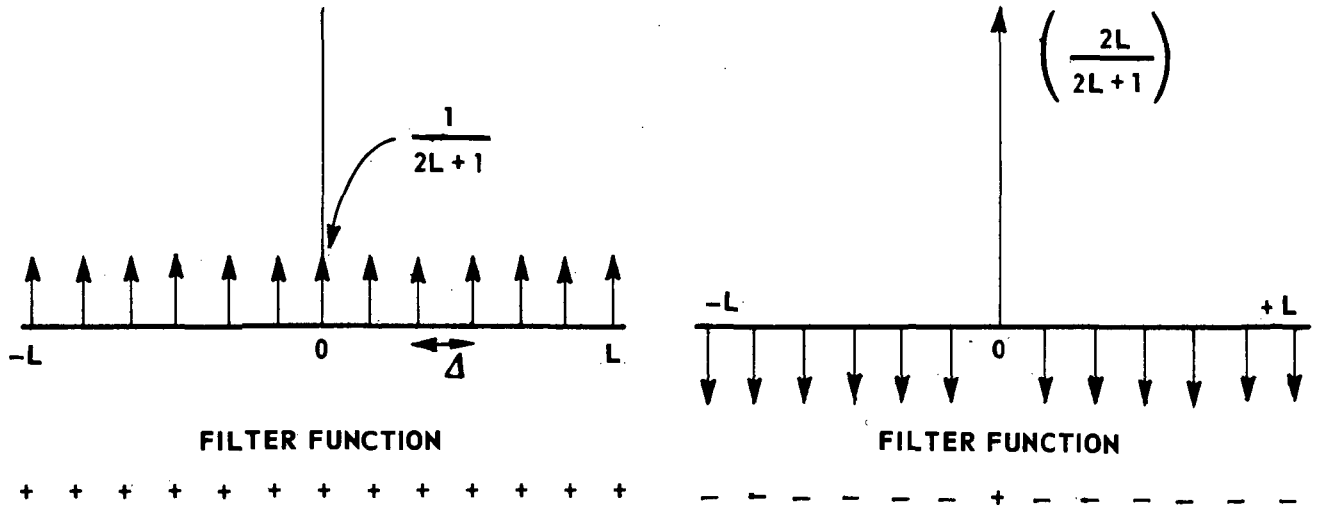


FIGURE 3.19 REPRESENTATION OF CONSTANT WEIGHT FILTER FUNCTIONS

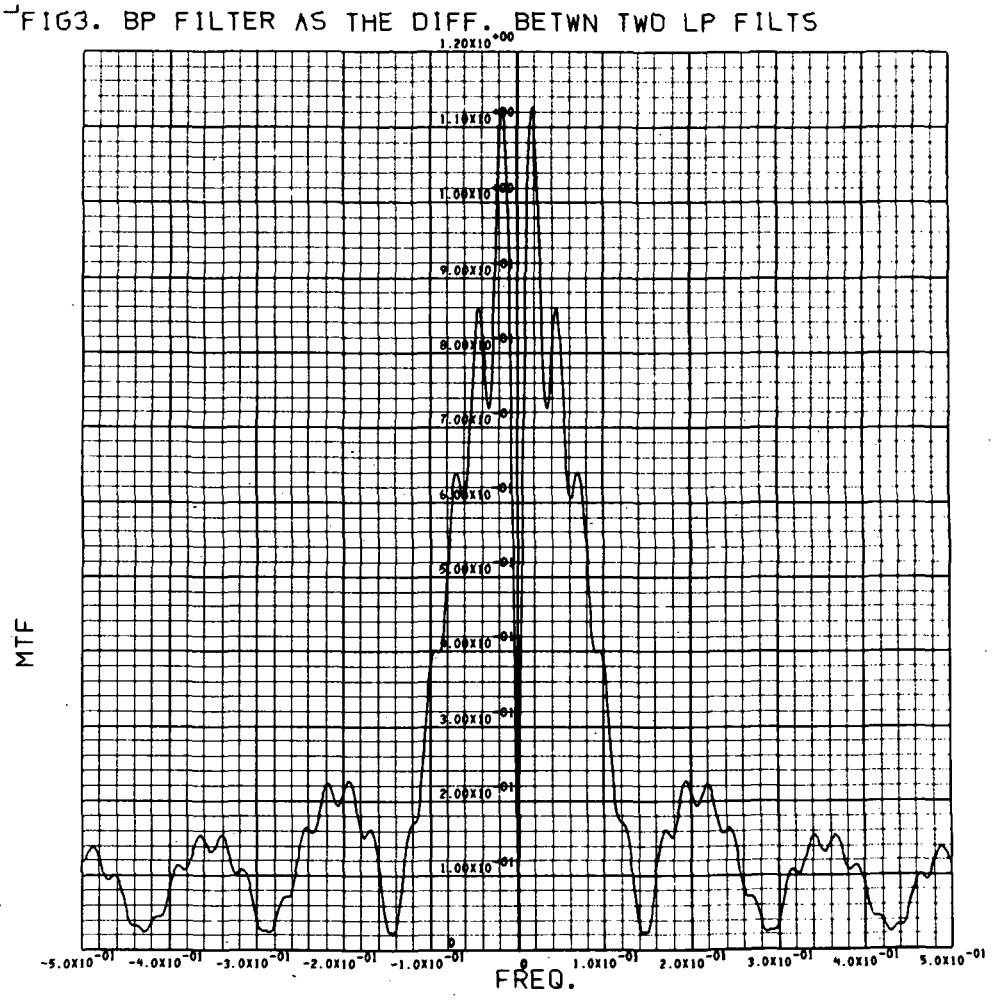


Figure 3.20 Band Pass Filter as the Difference between two Low Pass Filters

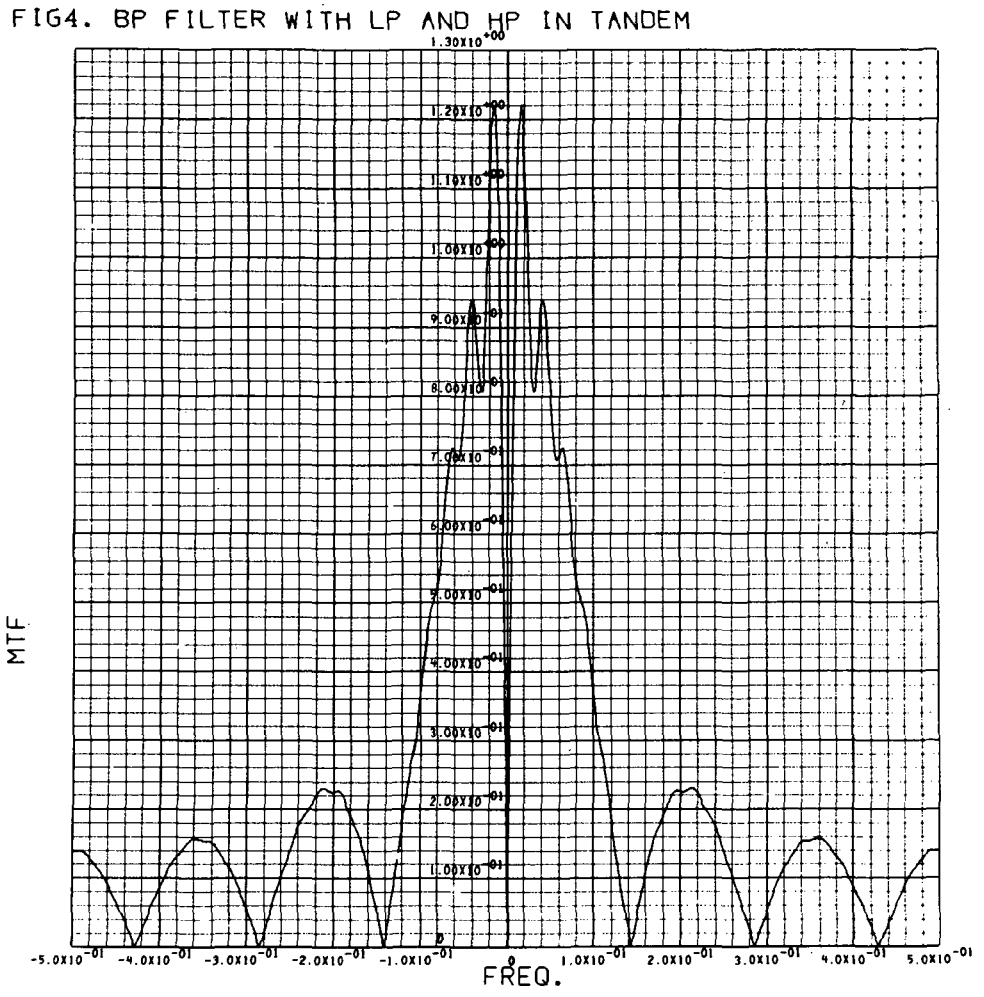


Figure 3.21 Band Pass Filter with Low Pass and High Pass Filters in Tandem

b. Filter realizations:

(1) Z - Transform representation of filter functions: A convenient way to obtain the recursive realization of a filter in general is to examine the Z-transform of its impulse response. If the impulse response (represented numerically by the filter weights) of a filter is given by the sequence $\{g_i\}$, then the Z-transform G(Z) is given by (see [19], for example)

$$G(Z) = \sum_{i=-\infty}^{\infty} g_i Z^{-i} \tag{3.48}$$

which is defined for all Z if $\{g_i\}$ has only a finite number of nonzero elements.

Now, the Z-transform of the output $\{y_i\}$ is related to the Z-transform of the input $\{x_i\}$ by

$$Y(Z) = G(Z)\ X\ (Z) \tag{3.49}$$

Therefore, from Equation (3.40), the input/output relationship for the low pass filter can be written as

$$Y(Z) = \frac{1}{(2L + 1)} \frac{(Z^L - Z^{-(L+1)})}{(1 - Z^{-1})} \quad X(Z) \tag{3.50}$$

The realization of this as a first order recursive filter is shown in Figure 3.22. Here Z^{-1} denotes unit delay. The Z-transform of the impulse response of the high pass filter of Equation (3.41) is given by

$$H(Z) = 1 - \frac{1}{(2L + 1)} \frac{(Z^L - Z^{-(L+1)})}{(1 - Z^{-1})} \tag{3.51}$$

The recursive realization corresponding to Equation (3.51) is shown in Figure 3.23. Here, LP(L) denotes the low pass filter shown in Figure 3.22.

It is easy to see that the band pass filters defined by Equations (3.45) and (3.47) can be realized as shown in Figures 3.24 and 3.25 respectively. Also, recursive realizations can be readily derived for some two dimensional low, high and band pass filters from the basic low pass filter realization in Figure 3.22.

(2) Two dimensional filters: Two general types of two-dimensional filters are realizable by a 'two pass' procedure, as discussed in the beginning of Section III-B: (i) "Sum" filters whose transfer function can be written as a sum of two functions each being a function of a single frequency variable; (ii) "Product" filter whose transfer function can be written as a product of two functions each being a function of a single frequency variable. "Sum" filters can be implemented by filtering the input separately in the horizontal and vertical directions and then adding the outputs. "Product" filters can be implemented by filtering the input in one direction and then filtering the result in the perpendicular direction. Some of these two-dimensional filters are discussed below.

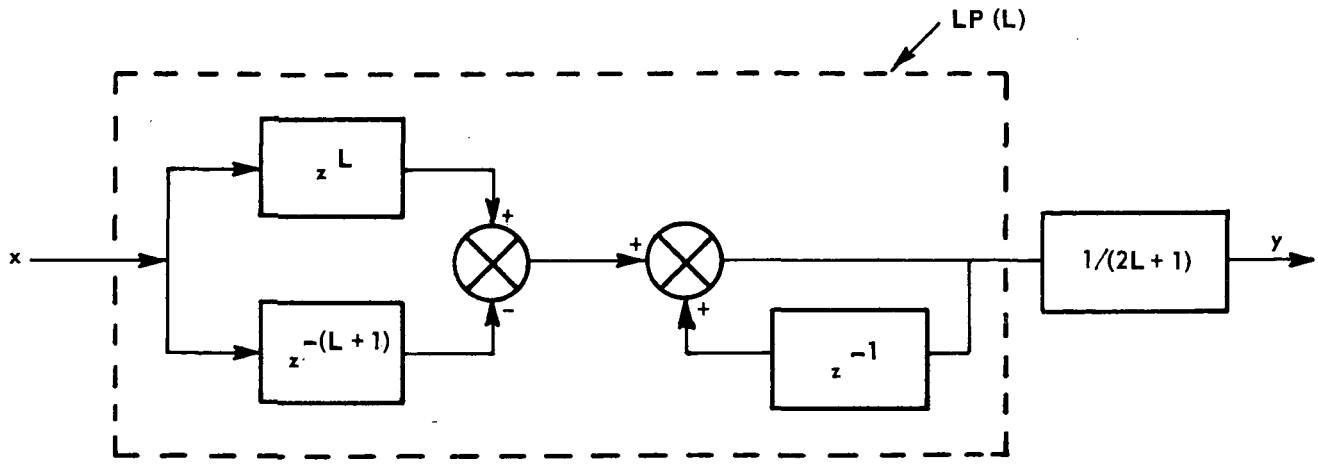


FIGURE 3.22 RECURSIVE REALIZATION OF THE LOWPASS FILTER OF EQUATION (3.40)

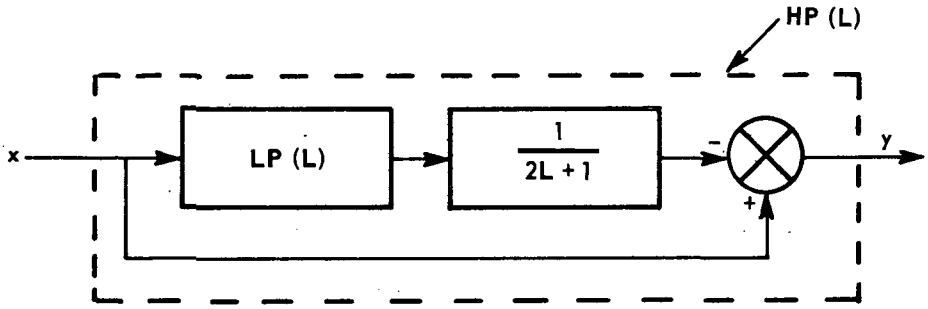


FIGURE 3.23 RECURSIVE REALIZATION OF THE HIGHPASS FILTER OF EQUATION (3.41)

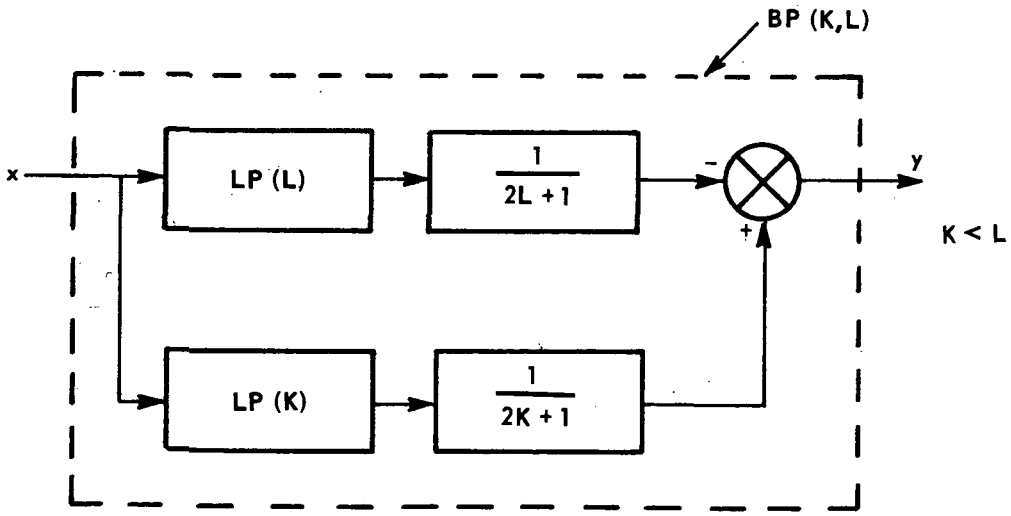


FIGURE 3.24 RECURSIVE REALIZATION OF THE BANDPASS FILTER OF EQUATION (3.45)

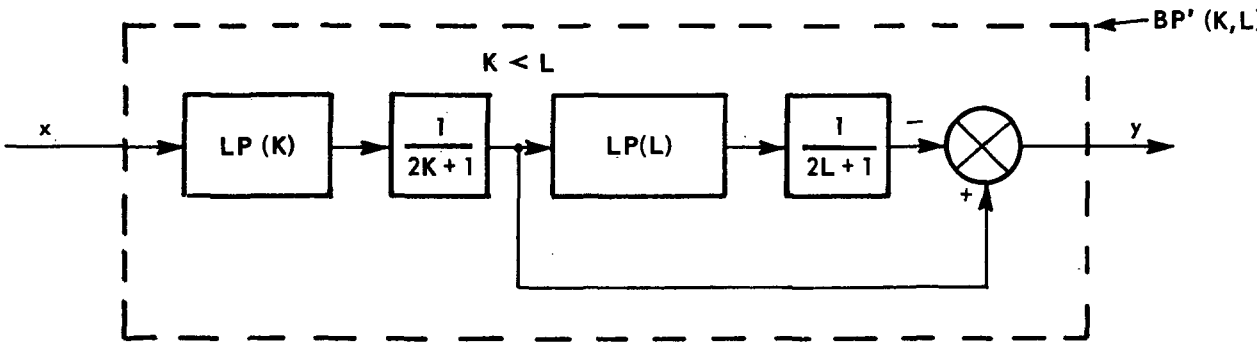


FIG. 3.25 RECURSIVE REALIZATION OF THE BAND PASS FILTER OF EQUATION (3.47)

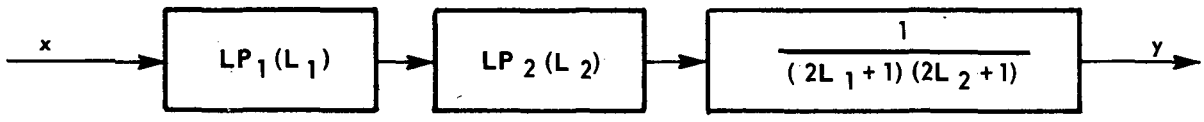


FIG. 3.26 RECURSIVE REALIZATION OF A 2 DIMENSIONAL LOW PASS FILTER (MOVING AVERAGE GENERATOR)

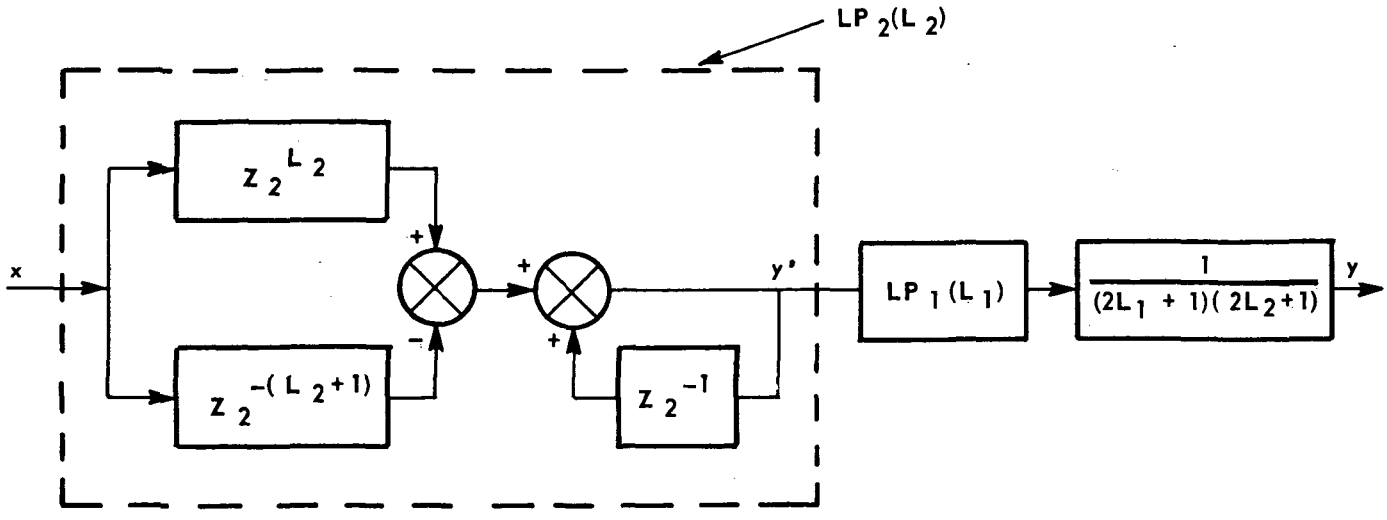


FIG. 3.27 ALTERNATE REPRESENTATION OF 2 DIMENSIONAL LOW PASS FILTER IN FIGURE 3.26

Suppose, for example, that it is necessary to implement a filter whose output corresponding to a given point in a picture is the average density of a $(2L_1+1)$ x $(2L_2+1)$ rectangle centered about the given point. Then, denoting basic low pass filters in the horizontal and vertical directions by $LP_1(.)$ and $LP_2(.)$ respectively, the above filter can be realized as shown in Figure 3.26. Of course, there is a difference in the operations of $LP_1(.)$ and $LP_2(.)$ in that $LP_1(.)$ handles points within a record (it operates along rows of the picture array) while $LP_2(.)$ handles points from several records (it operates down columns of the picture array). Table 3.4 shows the realizations of a number of two-dimensional filters using the basic low pass filter of Figure 3.22. The table also shows the nature of the point spread function of each filter, represented by the signs of the weights. In the implementation of the filter shown in Figure 3.26, for example, the tape-to-core and core-to-tape transfers and tape rewinds can be minimized by having the data to be filtered on two tapes and reading the appropriate records from each tape and then operating $LP_1(.)$ on them. These operations are represented in Figure 3.27. Here Z_2^{-1} is a unit delay in the vertical direction. Thus, when the i^{th} record is being filtered, the $(i + L_2)^{th}$ and $(i - L_2 - 1)^{st}$ records are read, each from a different tape, and their difference is added to the $(i - 1)^{st}$ record of y' which is held in core. $LP_1(L_1)$ is then operated on y'.

(3) Practical aspects of filter implementation: The manner of implementing the recursive realization of the filters summarized in Table 3.4 is dependent entirely on the medium available for storage of the image data. If sufficient core storage is available to store the entire image array, the entire filter functions may be implemented as algorithms.

More realistically, the image array is stored on drum, disk or tape, and to avoid excessive data seeking, the algorithms are modified slightly and the image data records are restructured to facilitate processing.

The Z-transform variable may be identified with the forward shift operator E associated with finite-difference equations. A filter operator Z^L may thus be interpreted as a forward shift of a data sequence by L sample intervals, and Z^{-L} may similarly be interpreted as a backward shift by L sample intervals (or correspondingly a delay of L units). When a data sequence is stored in its entirety in core, a forward shift with respect to a specific sample may be realized almost instantaneously, but this is not feasible when the required data sequence is held within records stored sequentially on tape. The forward shift operation is then implemented physically by duplicating the data records on tape and advancing the supplementary tape by the appropriate shift interval.

It is found in general that the number of arithmetic operations is reduced if the vertical filtering is done first. The number of tape rewinds is minimized by having the data on several tape units. The number of tape units on which input data is to be present depends on the number of input records needed simultaneously to form one record of output.

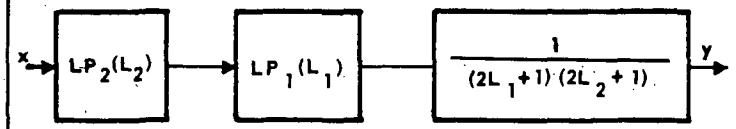
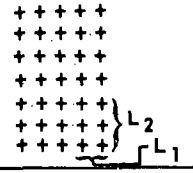
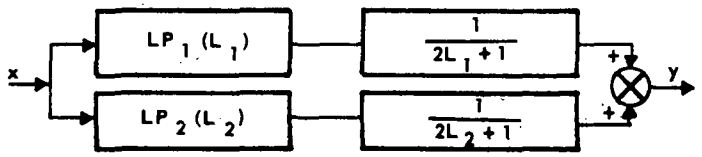
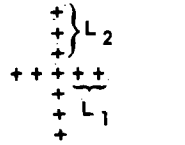
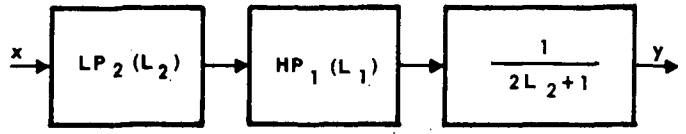
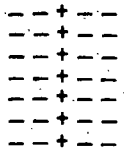
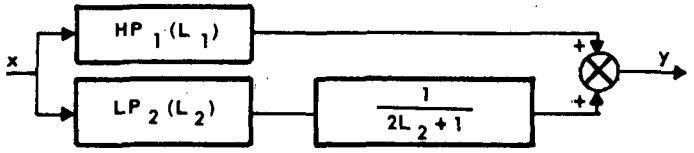
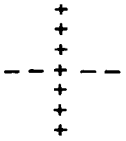
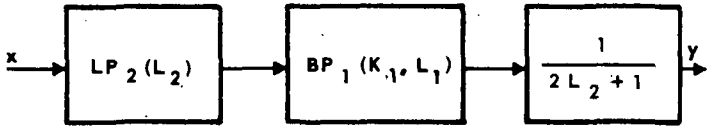
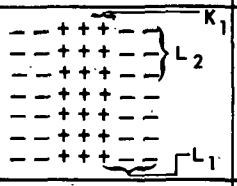
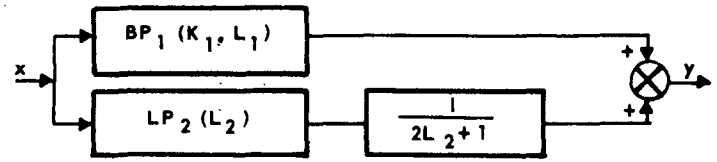
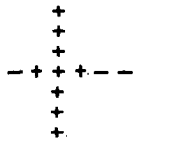
FILTER			REALIZATION	POINT SPREAD FUNCTION (FILTER WEIGHTS)	NO. OF INPUT TAPES	
HORIZ.	VERT.	TYPE			DATA	DATA & REFLECTION
L.P.	L.P.	PRODUCT			1	1
L.P.	L.P.	SUM			2	1
H.P.	L.P.	PRODUCT			1	1
H.P.	L.P.	SUM			2	1
B.P.	L.P.	PRODUCT			1	1
B.P.	L.P.	SUM			2	1

TABLE 3.4 RECURSIVE FILTER REALIZATIONS

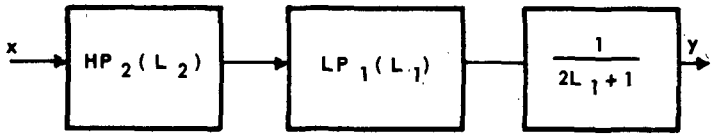
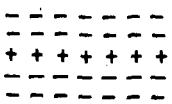
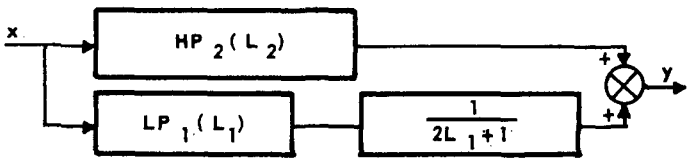
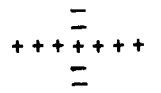
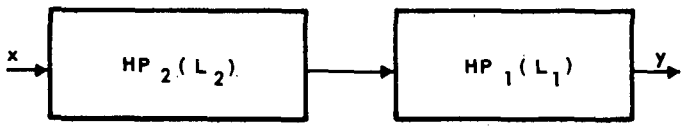
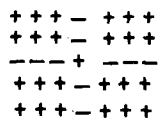
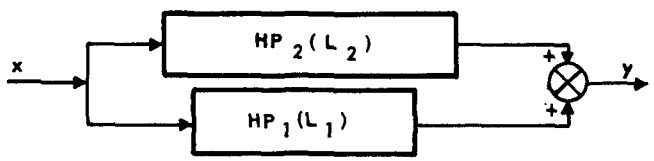
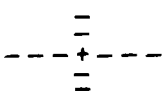
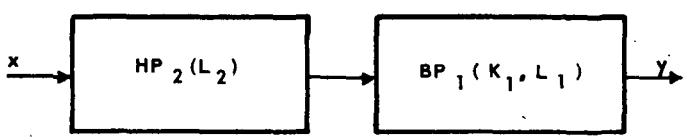
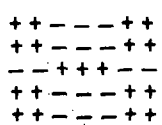
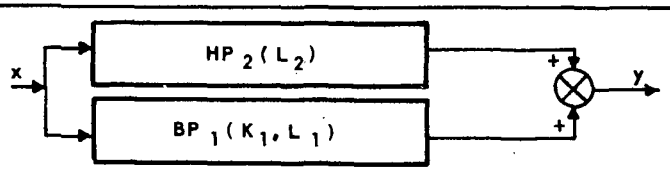
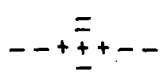
FILTER			REALIZATION	POINT SPREAD FUNCTION (FILTER WEIGHTS)	NO. OF INPUT TAPES	
HORIZ.	VERT.	TYPE			DATA	DATA & REFLECTION
L.P.	H.P.	PRODUCT			2	1
L.P.	H.P.	SUM			2	1
H.P.	H.P.	PRODUCT			2	1
H.P.	H.P.	SUM			2	1
B.P.	H.P.	PRODUCT			2	1
B.P.	H.P.	SUM			2	1

TABLE 3.4 RECURSIVE FILTER REALIZATIONS

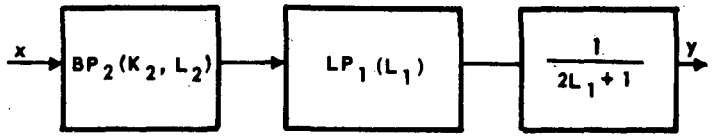
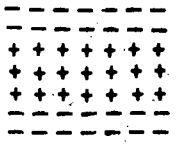
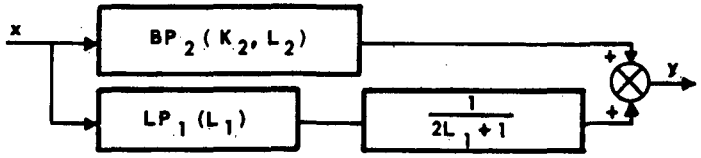
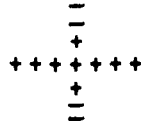
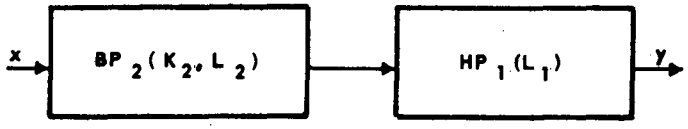
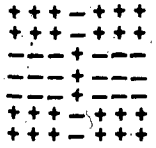
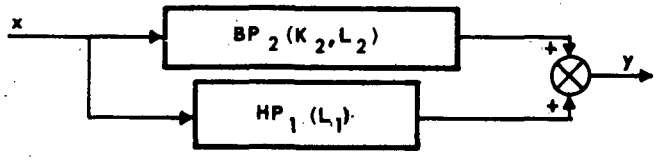
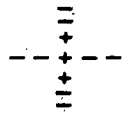
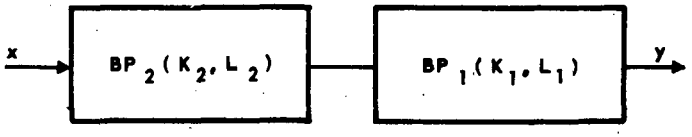
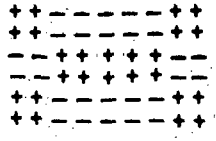
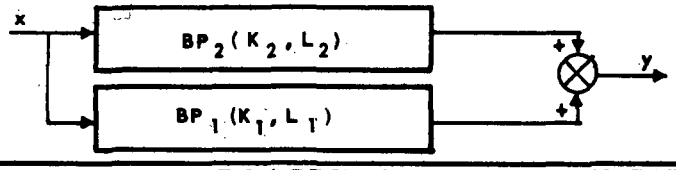
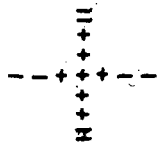
FILTER			REALIZATION	POINT SPREAD FUNCTION (FILTER WEIGHTS)	NO. OF INPUT TAPES	
HORIZ.	VERT.	TYPE			DATA	DATA & REFLECTION
L.P.	B.P.	PRODUCT			2	2
L.P.	B.P.	SUM			3	2
H.P.	B.P.	PRODUCT			2	2
H.P.	B.P.	SUM			3	2
B.P.	B.P.	PRODUCT			2	2
B.P.	B.P.	SUM			3	2

TABLE 3.4 RECURSIVE FILTER REALIZATIONS

Also, the edges of the picture should be augmented properly in order to avoid unpleasant edge effects. This can be done by defining the data outside the domain of the picture arbitrarily (See Section III-B.1). It has been found from previous experience that good visual effects are obtained by defining the data outside the domain to be the reflection of the data in the nearest edge. The reflections of data needed while filtering each record (i.e., horizontal filtering) can be generated in core. But the reflections needed for vertical filtering have to be stored on tape. The number of records which should be reflected at the top (and bottom) of the picture is K where (2K + 1) is the maximum anticipated vertical filter length. In fact, it is convenient to handle the records if a tape is prepared with the reflection at the top appearing first, the data records themselves appearing next and the reflection at the bottom appearing at the end.

In Table 3.4, the number of tape units on which input data should be supplied and also the number of tape units on which data and reflections should be supplied are shown for each of the filters.

c. Extension to matched filter implementation: We shall now present a simple extension of the recursive filters discussed above to some particular cases of matched filters. We shall restrict ourselves to cases where only the variations of density over the local average are of interest. We can therefore choose the weights $g_{k\ell}$ in equation (3.26) such that

$$\sum_{k,\ell} g_{k\ell} = 0 \tag{3.52}$$

(this condition is satisfied by filters which are not low pass at least in one dimension). Also, equation (3.26) is now modified to

$$y_{ij} = \frac{\sum_{k=-K}^{K} \sum_{\ell=-L}^{L} g_{k\ell} (x_{i+k,j+\ell} - \bar{x}_{ij})}{\sqrt{\sum_{k,\ell} g_{k\ell}^2 \sum_{k,\ell} (x_{i+k,j+\ell} - \bar{x}_{ij})^2}} \tag{3.53}$$

where

$$\bar{x}_{ij} = \frac{1}{(2K+1)(2L+1)} \sum_{k,\ell} x_{i+k,j+\ell} \tag{3.54}$$

Now, from (3.52) it follows that the numerator of (3.53) is

$$\sum_{k=-K}^{K} \sum_{\ell=-L}^{L} g_{k\ell} x_{i+k,j+\ell}$$

Also, from (3.54), it is easy to see that

$$\sum_{k,\ell} (x_{i+k,j+\ell} - \bar{x}_{ij})^2 = \sum_{k,\ell} x_{i+k,j+\ell}^2 - (2K+1)(2L+1)\bar{x}_{ij}^2 \tag{3.55}$$

Further, note that $\Sigma_{k,\ell} \; g_{k\,\ell}^2$ is a constant over all i,j and hence if linear scaling will be used for displaying data (as is most often the case) this factor can be omitted from the expression (3.53) for y_{ij}.

Therefore, the recursive filters described above can be used to generate y_{ij} with the modification shown in Figure 3.28. Note that in all the filters shown in Table 3.4, $LP_2(L_2)$ appears either explicitly or implicitly and that in some cases the tandem combination $LP_2(L_2)LP_1(L_1)$ on x in Figure 3.28 can be eliminated. Also, no additional input/output operations are involved in the computation of the matched filter output given by (3.53) compared with the computation of only the numerator of (3.53) as is done in the case of the filters in table 3.4. Thus we have an efficient implementation of matched filters which can be used for detection of shapes resembling the point spread functions in table 3.4.

Examples of matched filtering of a test pattern using the above method are shown in figures 3.29, 3.30 and 3.31. The basic test pattern used in these figures consists of vertical lines of high density (density number = 15) separated by areas of low density (density number = 0). The vertical lines have widths 1, 5, 9, ..., 25 pixels and the areas separating them have widths of 25 pixels. The template used in all these pictures is a 39 x 49 pixel rectangle with a vertical line of width 13 pixels at the center.

In figure 3.29, the frame in the top left corner shows the test pattern rescaled between 0 x 63. The second frame in the first column is the result of matched filtering which is linearly rescaled so that the minimum is set to 0 and the maximum is set to 63. The remaining frames are all results of rescaling the filtered output with progressively increasing thresholds, by setting values below the threshold to 0 and linearly scaling the values above the threshold to lie between 0 and 63.

In figure 3.30, the frame in the top left corner shows the unfiltered test pattern corrupted with additive noise which is approximately white and Gaussian with a mean of 15 and variance of 8. The noisy pattern is rescaled linearly to lie between 0 and 63. The second frame and all the other frames are the results of matched filtering, thresholding and rescaling as in figure 3.29.

Note that as the threshold is increased in figures 3.28 and 3.29, the vertical lines get thinner and thinner, the lines farther away from the fourth vertical line (i.e. the line of width 13 pixels) disappear first and in the last (bottom righthand corner) frame only the line corresponding to the fourth line remains. This clearly demonstrates the effect of mismatch upon the matched filter output.

Figure 3.30 shows the effect of varying the input noise level on the matched filtered output. The left column shows the test pattern corrupted with increasing noise levels as one proceeds downward. The right column shows the corresponding matched filtered outputs which have been thresholded and rescaled. The threshold is taken as 0. That is, all output values

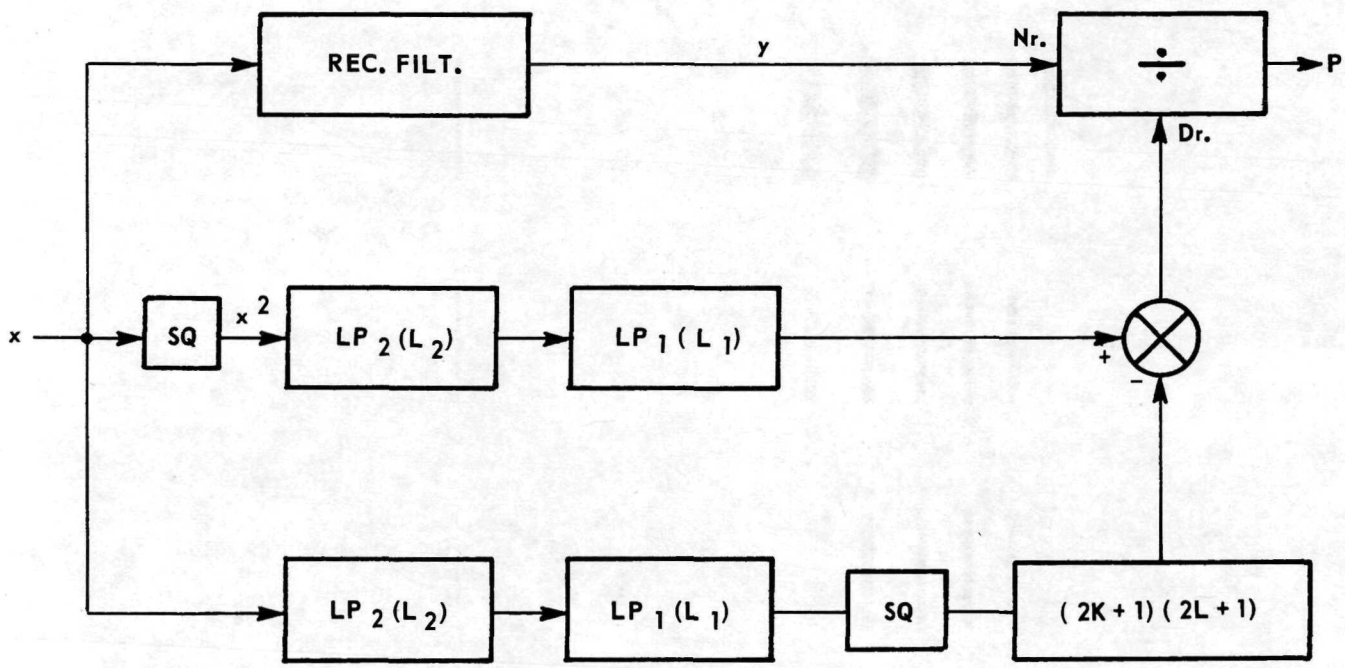


FIGURE 3.28 MATCHED FILTER REALIZATION

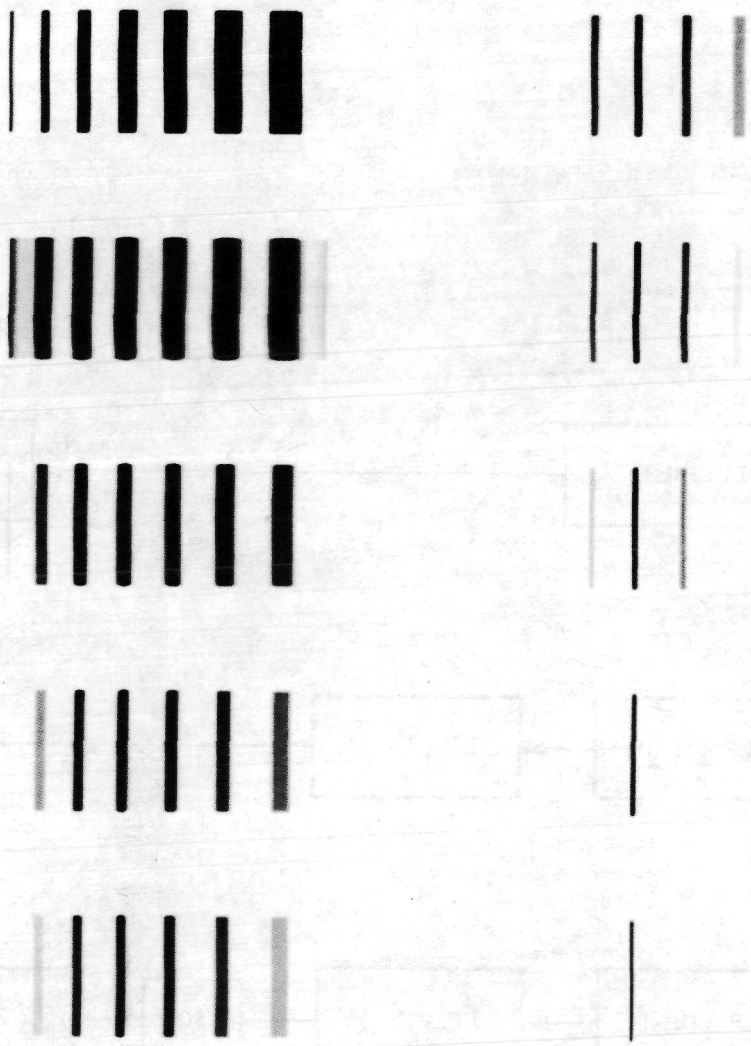


Figure 3.29 Test Pattern and Matched Filter Output with Different Thresholds

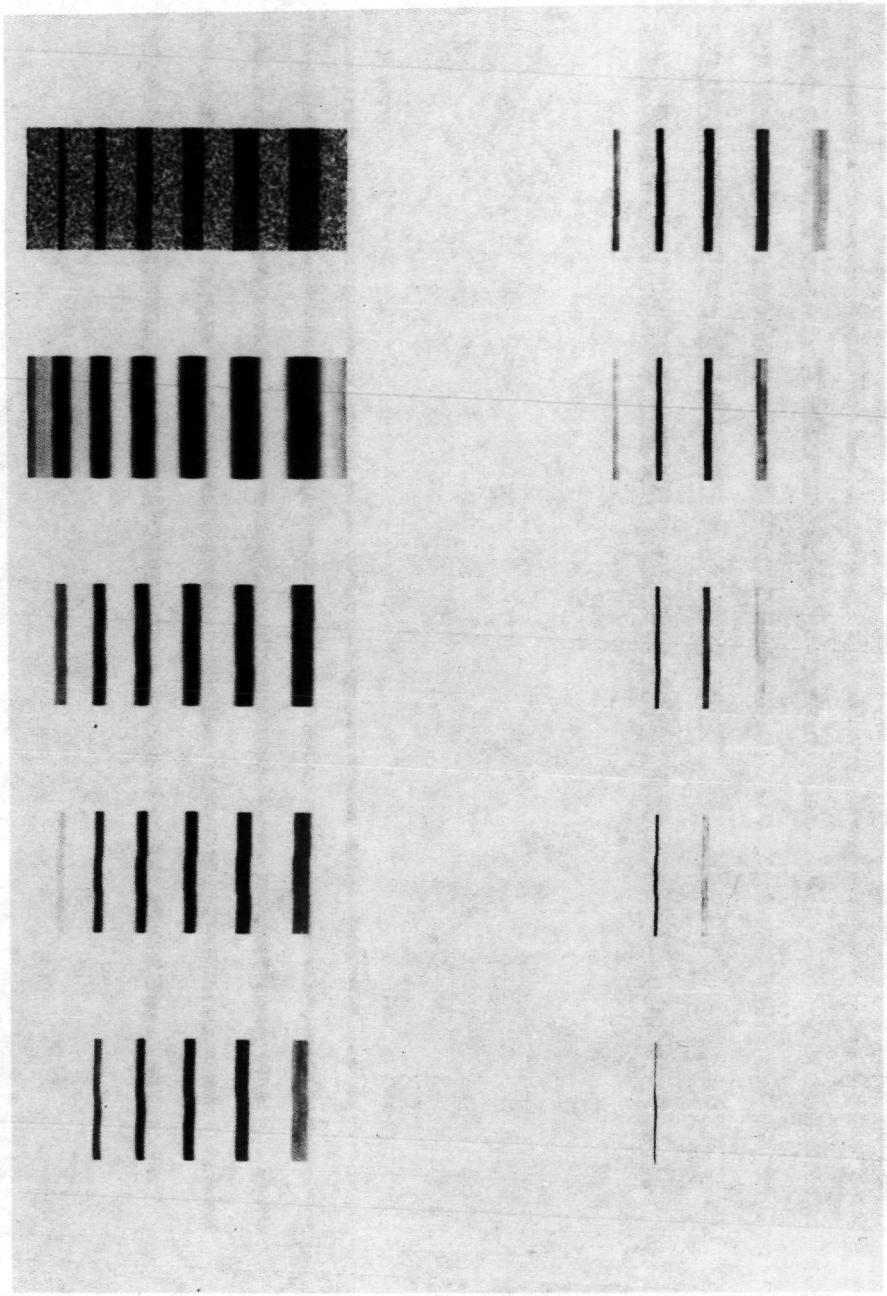


Figure 3.30 Noisy Test Pattern and Matched Filter Output With Different Thresholds

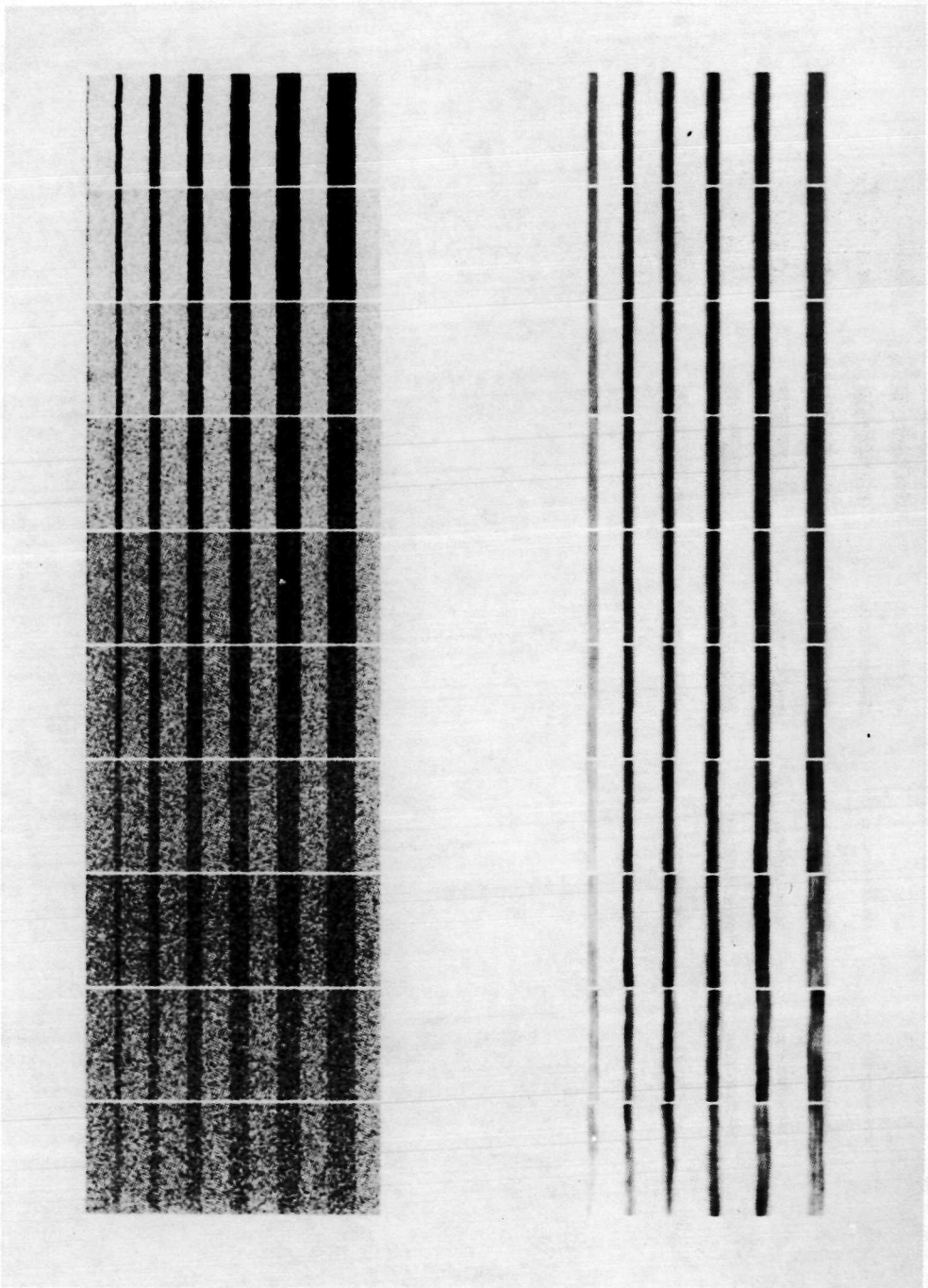


Figure 3.31 Noisy Test Patterns with Different Noise Levels Before and After Matched Filtering

less than 0 have been set to 0 and the nonnegative values have been linearly rescaled to lie between 0 and 63. It can be noted that the filtered pictures are reasonably good replicas of the original (noise free) picture except that in the high noise cases (last two) there is some noticeable distortion.

f. Comparison of implementation times: It might be of some interest to compare the times taken to filter a picture using nonrecursive implementation using fast convolution and the recursive implementation of the particular filters discussed above. Of course, it is to be remembered that the recursive implementation discussed in Section III-B.2 is only for particular transfer functions and the nonrecursive implementation in Section III-B.1 can implement any transfer function. Table 3.5 shows the typical values of the time needed to filter a 256 x 256 picture in one dimension using different lengths of filters.

Filter Length	Time (Min.)
40	15
150	40
200	84
240	85

Table 3.5 Times for Implementation of a General One-Dimensional Filter (Nonrecursive)

These times can be reduced slightly if the fact that the data and the filter weights are real is exploited. However, their orders of magnitude will be retained. Further, if two-dimensional filtering is needed, a two pass procedure has to be employed (under the present core limitations) involving two large matrix transpositions and two one-dimensional filtering operations. Consequently, the times shown in Table 3.5 will be more than doubled for two dimensional filtering.

In the case of radiographic images in which fine features such as minute cracks are to be detected it is necessary to digitize the imagery at the highest avilable resolution, viz., 12.5μ sampling interval. This means that even for an image size of 12.5mm x 12.5mm a 1000 x 1000 digital array is generated. Therefore, the nonrecursive filters mentioned above become rather impractical.

Experiments performed on a radiographic image of size 1000 x 320 and a filter which was band pass in the horizontal direction and low pass in the

vertical direction (see table 3.4) with parameters $k_1 = 6$, $L_1 = 24$ and $L_2 = 19$ have shown that the time taken for recursive implementation in two dimensions was of the order of 8.5 minutes which is a considerable reduction from the times shown in table 3.5. With the same filter and picture sizes the matched filter implementation took about 14.2 minutes.

3. Sequential similarity detection [20]: A promising new technique for the "automatic determination of the local similarity between two structured data sets" has been reported by Barnea and Silverman. They also illustrate the application of this technique for registration of remotely sensed imagery. This technique, called Sequential Similarity Detection works on the principle that similarity between an m x n template and an area on a picture can be defined in terms of an error criterion which is a sum of mn positive terms, each of the terms being the error between the template and the picture area at a particular point. Thus, the error is a monotonic nondecreasing function of the number of points at which it has been evaluated and summed. It increases slowly when the template matches the picture area approximately and increases rapidly if the template does not match the picture area. Thus, if a threshold is properly chosen on the error, one can decide whether a match occurs or not without actually computing the error completely for all the points. This is particularly advantageous in cases where the number of points at which the match occurs is small compared to the total number of points in the picture. Barnea and Silverman claim that this method is about 50 times faster than even FFT methods of correlation for registration.

The applications of this technique to image enhancement are being tried at present. Experiments are being performed on test patterns to develop criteria for threshold selection so that the processing time is minimized and the probability of detection of desired features in a corrupted image is maximized. The results of these experiments will be reported in subsequent reports.

SECTION IV. APPLICATIONS

The image processing techniques and software discussed in the previous sections have been applied to several radiographs. It is the purpose of this section to demonstrate these applications. Discussed are: (i) the enhancement of flaws in the radiograph of a structural weld and (ii) the enhancement of the radiograph of a metal with a crack in it in order to determine the extent and shape of the crack.

A. Structural Weld Radiograph

Figure 4.1 shows the radiographic image of a structural weld. This image was prepared by first digitizing the radiograph with 25μ scanning interval and then writing it on film using the Photowrite and printing the resulting image. While there are three flaws which are clearly visible (one large approximately circular dark spot, one approximately elliptic dark spot and a small spot about 1/2" to the right of it) it is not clear whether there are any cracks joining these flaws. The purpose of enhancement was to make the shapes of the flaws more evident and to detect cracks, if any, between the flaws.

A band emphasis filter was applied to the image in figure 4.1 so that edges and cracks would be enhanced but the high frequency noise would be suppressed. The transfer function used was the one shown in figure 3.6 except for the fact that the gain in the attenuated bands was set to .1 instead of 0. Figure 4.2 shows the effect of one dimensional filtering of the region containing the flaws in figure 4.1. The filtering was horizontal so that the vertical (and almost vertical) edges of the flaws have been enhanced. Figure 4.3 shows the effect of two dimensional filtering of the same region. The filter was implemented as a two pass procedure, as illustrated in figure 3.7. The transfer function was the same band emphasis function as above in both the horizontal and vertical directions. In this case both the horizontal and vertical edges are enhanced. Note the difference between the edges of the two larger flaws in figures 4.2 and 4.3 and in particular the fact that the nearly horizontal edge of the oval flaw is marked much better in figure 4.3.

The above filtered images show the shapes of the flaws quite clearly and indicate no crack joining the two larger flaws. However, there is some doubt as to whether the oval flaw and the small flaw to its right are joined by a hairline crack. In order to confirm the existence (or otherwise) of a crack, the region including the oval flaw and the small flaw was digitized with 12.5 scanning interval. The scanned area is shown in figure 4.4. The scanning direction in this case is perpendicular to that in figure 4.1 and the oval flaw appears vertically above the small flaw (instead of on the left). Figure 4.5 shows the effect of contrast enhancement by point operations of linear density stretching. (see section III-A.2.b). A 256 x 256 pixel area of figure 4.4 containing the flaws of interest was

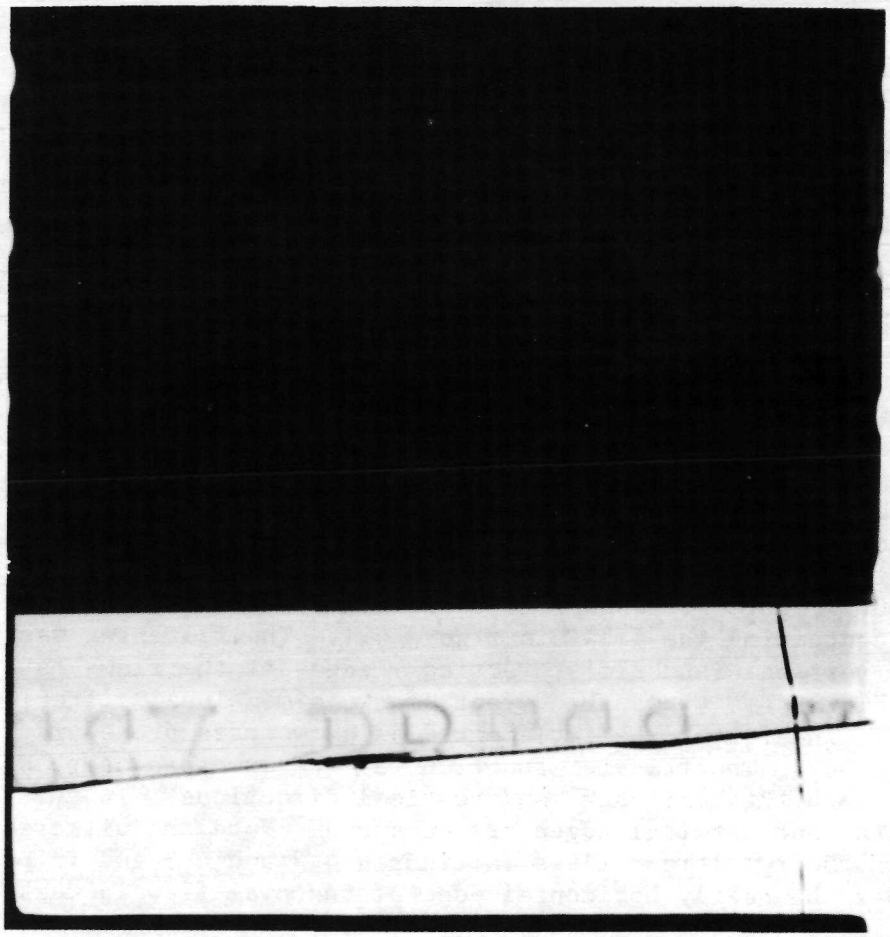


Figure 4.1 Radiographic Image of a Structural Weld

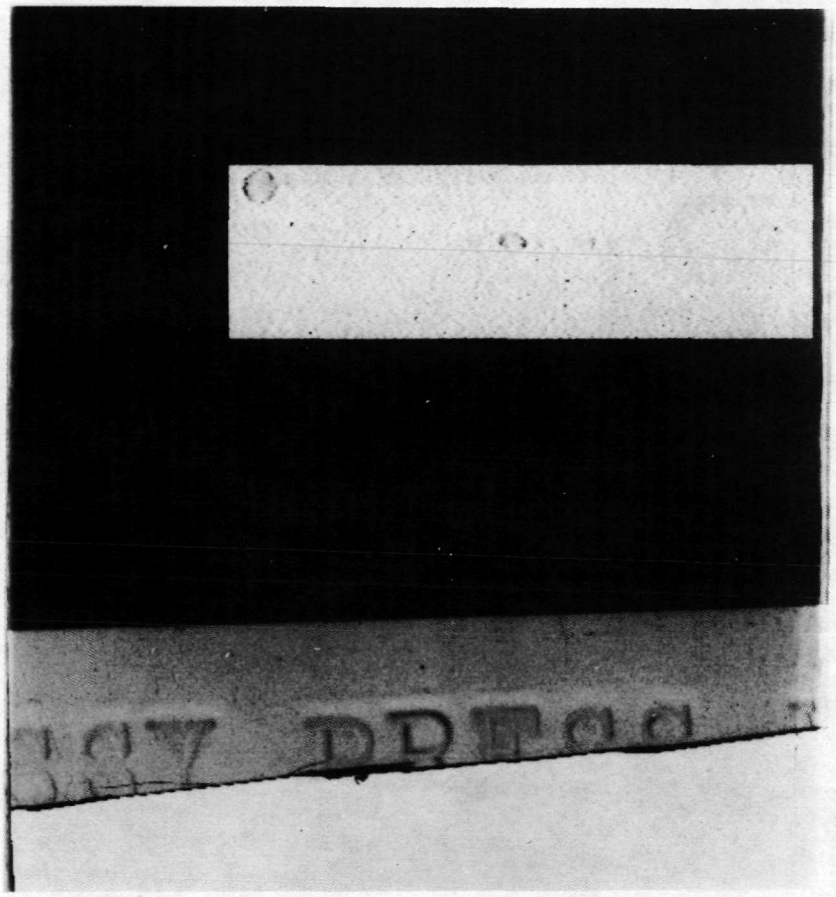


Figure 4.2 Image after Horizontal Band Emphasis Filtering of Part of Figure 4.1

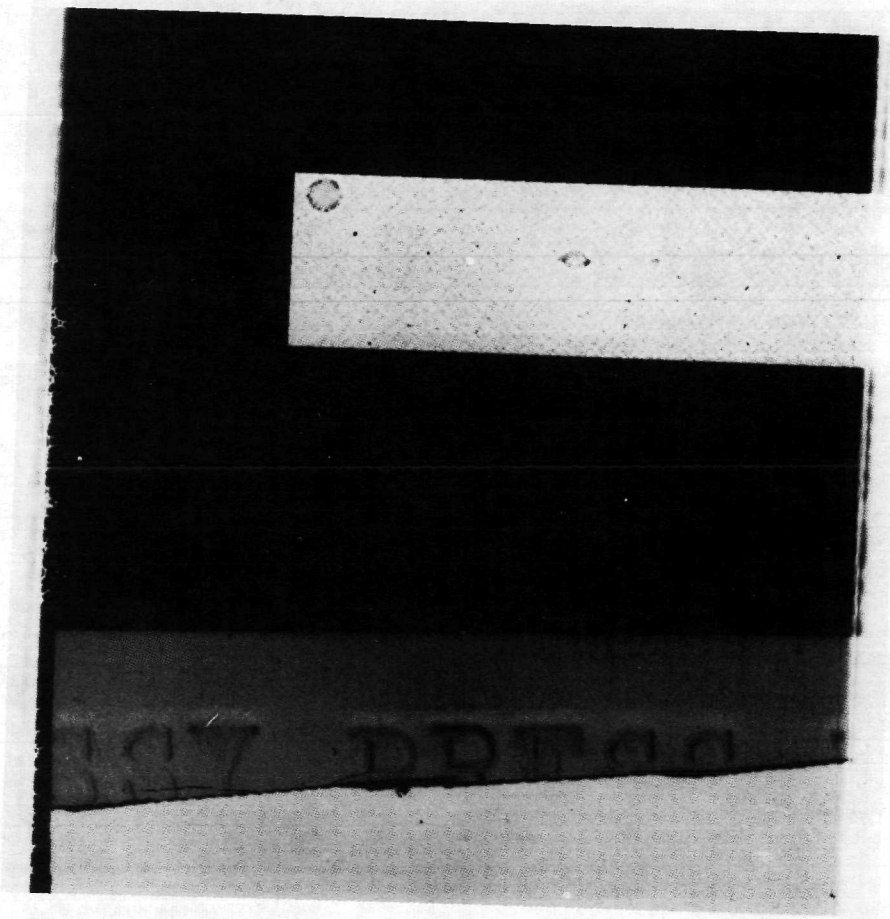


Figure 4.3 Image after Two Dimensional Band Emphasis Filtering of Part of Figure 4.1

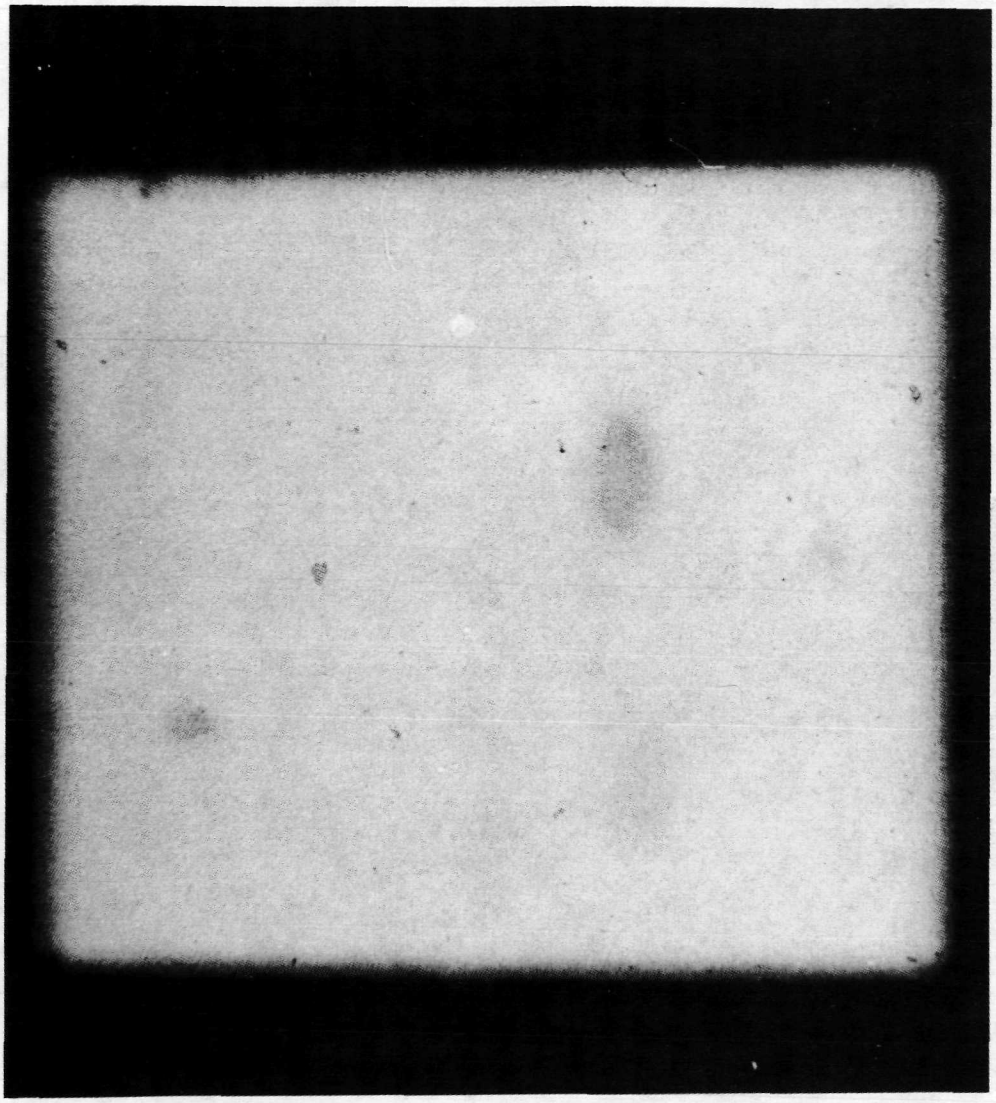


Figure 4.4 Small Part of Figure 4.1 Digitized with 12.5 μ Scanning Interval

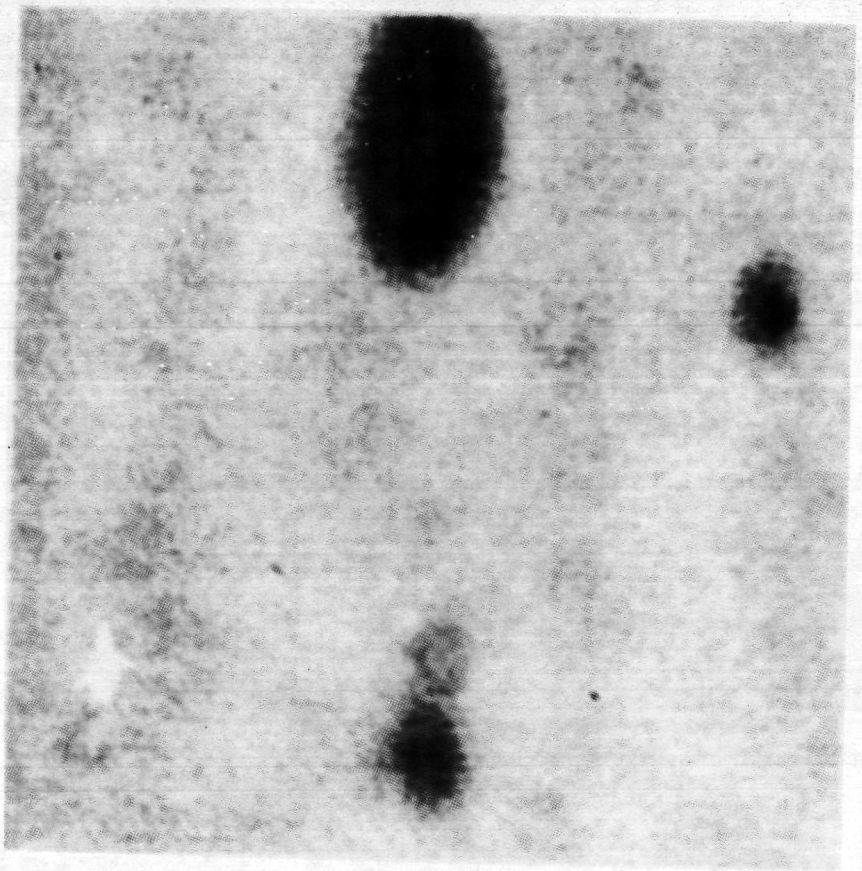


Figure 4.5 Image after Linear Density Stretching was Applied to Part of Figure 4.4 for Contrast Enhancement

extracted and the minimum density number in the area was subtracted from the density number at each point and the resulting numbers were linearly rescaled between 0 and 63. Enlargement by a factor of 3 in each direction was obtained by repeating each pixel density three times in both the horizontal and vertical directions. The flaws are very clearly visible in figure 4.5 and this figure indicates that there is no hairline crack joining the two flaws. This area was further explored by one dimensional and two dimensional filters having several different transfer functions. Figure 4.6 shows six cases of one dimensionally high pass filtered images. The filtering in all the cases was in the horizontal direction. The implementation of the filters was recursive as discussed in section III-B.2 and the lengths of the filters for the six cases are 3, 23, 43, 63, 83 and 103 pixels respectively. The filtered data was linearly rescaled between 0 and 63 in all cases. The first three cases are shown in the first column and the last three in the second column of figure 4.6. The filter length increases with each frame as one proceeds downward in the figure. It can be seen that the smaller the filter length, the sharper are the vertical edges of the flaws except in the case of filter length 3 (top left frame) where the high frequency noise is predominant. Figure 4.7 shows the effect of two diminsional high pass filters of various sizes on the same region as in figures 4.5 and 4.6. A 1 x 1 'filter' was also attempted just as check on the program. This corresponds to subtracting the density at each pixel from itself and hence, as expected, the top left corner of figure 4.7 consists of a 'frame' with all densities equal to 0. The other five frames correspond to filter sizes of 11 x 11, 21 x 21, 31 x 31, 41 x 41 and 51 x 51 respectively, the order of frames being the same as in figure 4.6. As in figure 4.6, the shorter filters yield sharper edges and in this case the comment applies to both horizontal and vertical edges of the flaws. From these pictures it can be seen that there are no hairline cracks between the flaws.

B. Detection of a Crack in a Metal from its Radiograph

Figure 4.8 shows the radiographic image of a metal. This image was obtained by scanning the radiograph with a 12.5μ interval and writing it back using the Photowrite and printing the image. The lighter rectangular area is the region of interest. A thin vertical crack can be seen to begin at the top of the picture. The crack is visible better in the radiograph than in the printed version of it. However, the end of the crack cannot be seen either in the original radiograph or the printed version. The purpose of enhancement was to find the extent of the crack by making the crack more prominent with respect to the background.

Figure 4.9 shows the result of subtracting the minimum density in figure 4.8 and stretching the densities between 0 and 63. It can be seen from this figure that the crack (the dark vertical line at the top of the picture) has become more prominent and merges into a dark patch on the right half of the picture (about 2.5" from the top) and extends beyond the patch sloping towards the right.

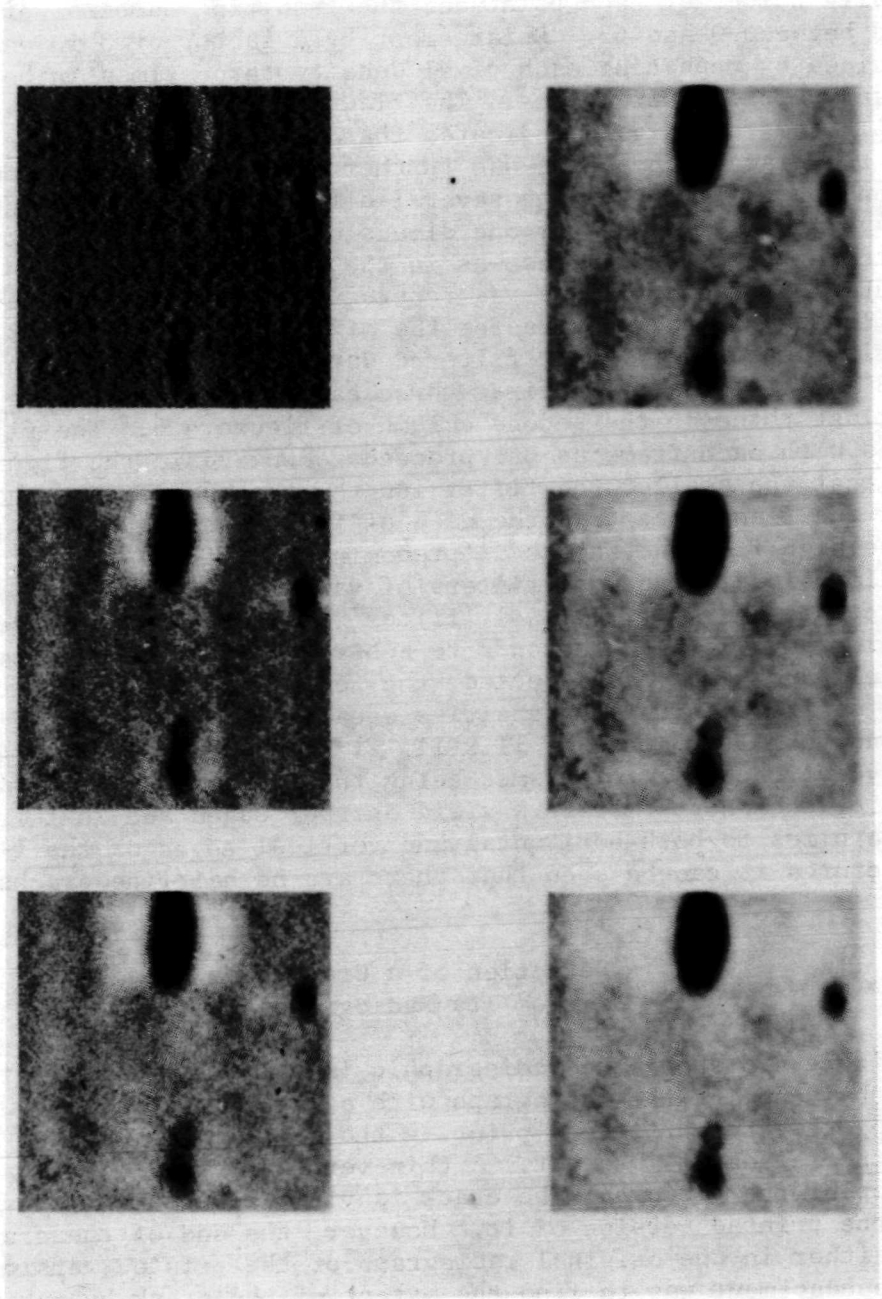


Figure 4.6 Effect of Various Lengths of High Pass Filter (Horizontal) on Figure 4.4

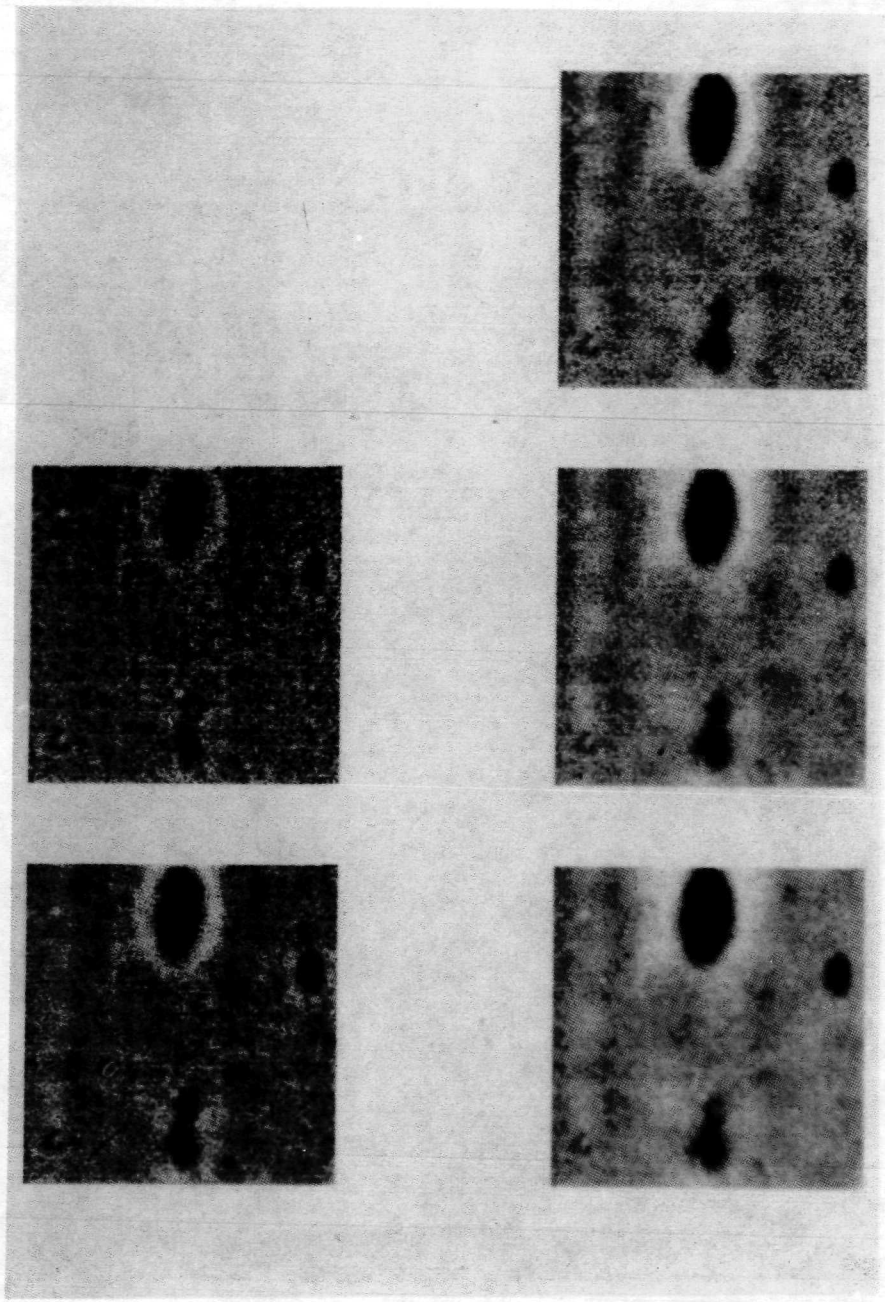


Figure 4.7 Effect of Various Sizes of High Pass Filter (Two Dimensional) on Figure 4.4

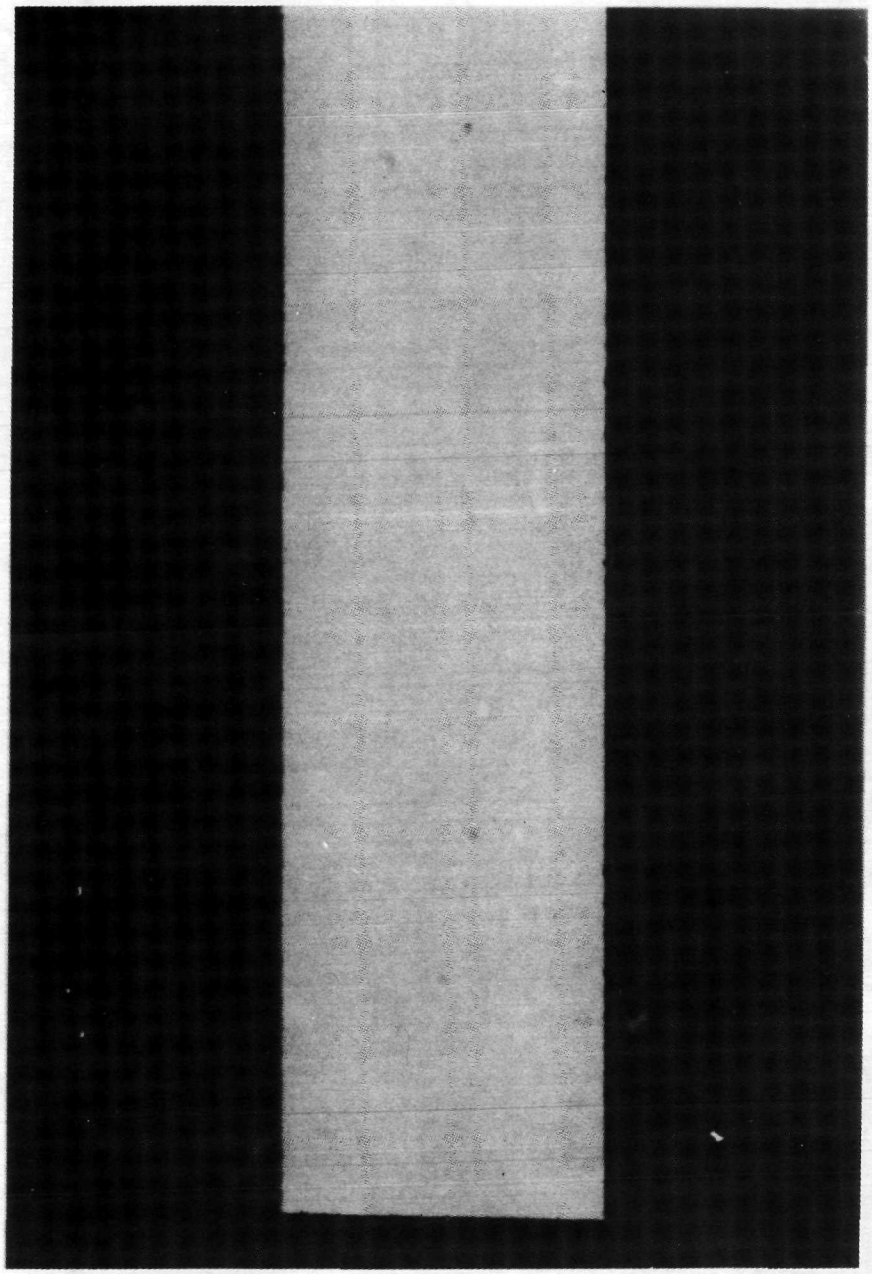


Figure 4.8 Radiographic Image of a Metal (I)

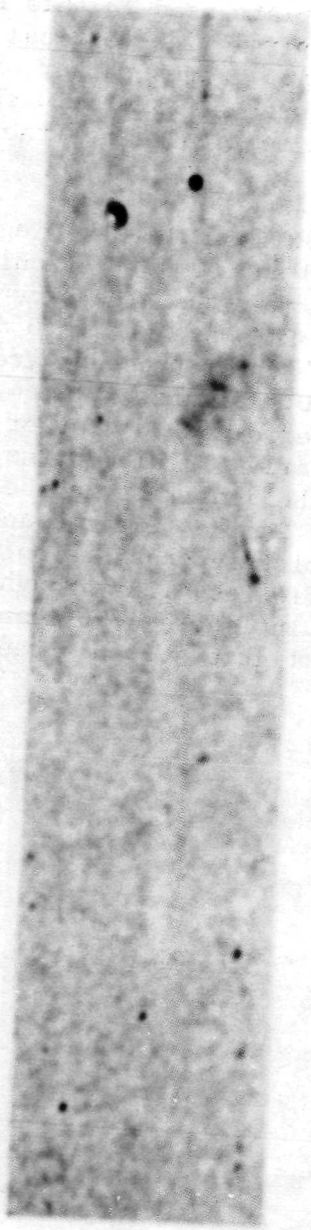


Figure 4.9 Linearly Rescaled (Density Stretched) Version of Figure 4.8

Figure 4.10 shows the result of applying a matched filter which enhances vertical lines to figure 4.9. The filter (described in section III-A.2.a) gives a high output for the positions in the picture which resemble a template. The template used in this case was a 39 x 49 pixel rectangle consisting of a 13 pixel wide vertical line in the center, so that the filter enhances vertical lines darker with respect to the background. A comparision of figures 4.9 and 4.10 shows that the crack in figure 4.9 yields a very dark vertical line (the darkest line beginning at the top of the picture).

Figure 4.11 is the result of thresholding the matched filter output before rescaling it between 0 and 63 so that only the regions of the picture which are, on the average, darker than the background along a vertical line are retained and the other points are set to 0 density. The position of the crack and the direction in which it proceeds are clearer in figure 4.11 than it is in figures 4.9 and 4.10. Figures 4.12, 4.13, 4.14 and 4.15 demonstrate the same set of operations performed on another radiograph. The crack is again more prominent and clearly visible in the filtered images. Figure 4.16 shows the effect of adding the densities in the rescaled version of figure 4.13 and twice the densities in the filtered and thresholded version of figure 4.15, point by point. This picture consists only of approximately $5/9^{ths}$ from the top of the pictures in figures 4.12 through 4.15 and has been enlarged. The crack is more prominent in figure 4.16 than in figure 4.13 and the background which was missing in figure 4.15 has been restored.

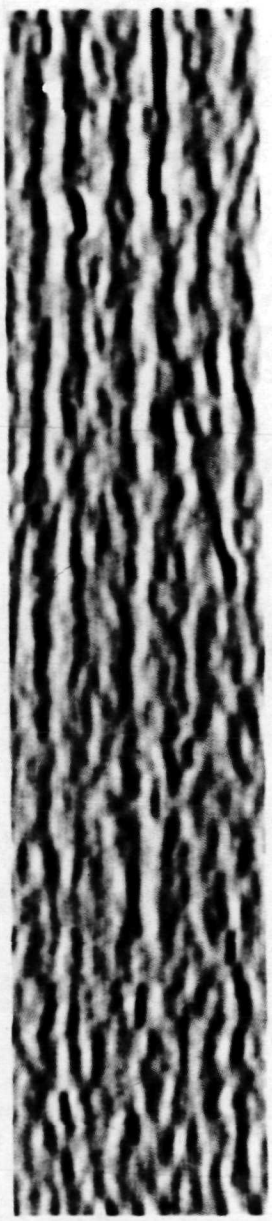


Figure 4.10 Result of Filtering Figure 4.9 with a Matched Filter to Enhance Dark Vertical Lines

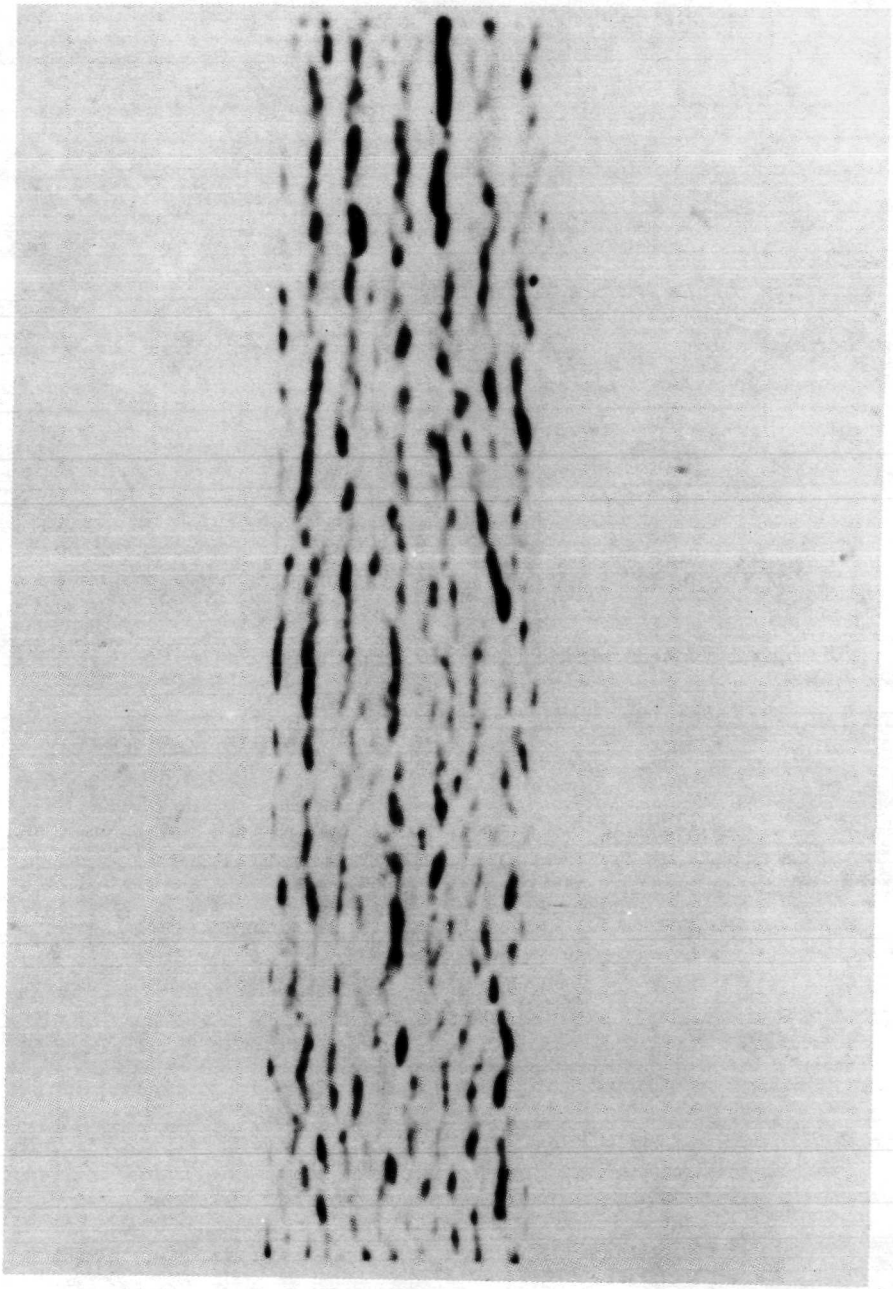


Figure 4.11 Result of Thresholding the Matched Filtered Output in Figure 4.10 Before Rescaling

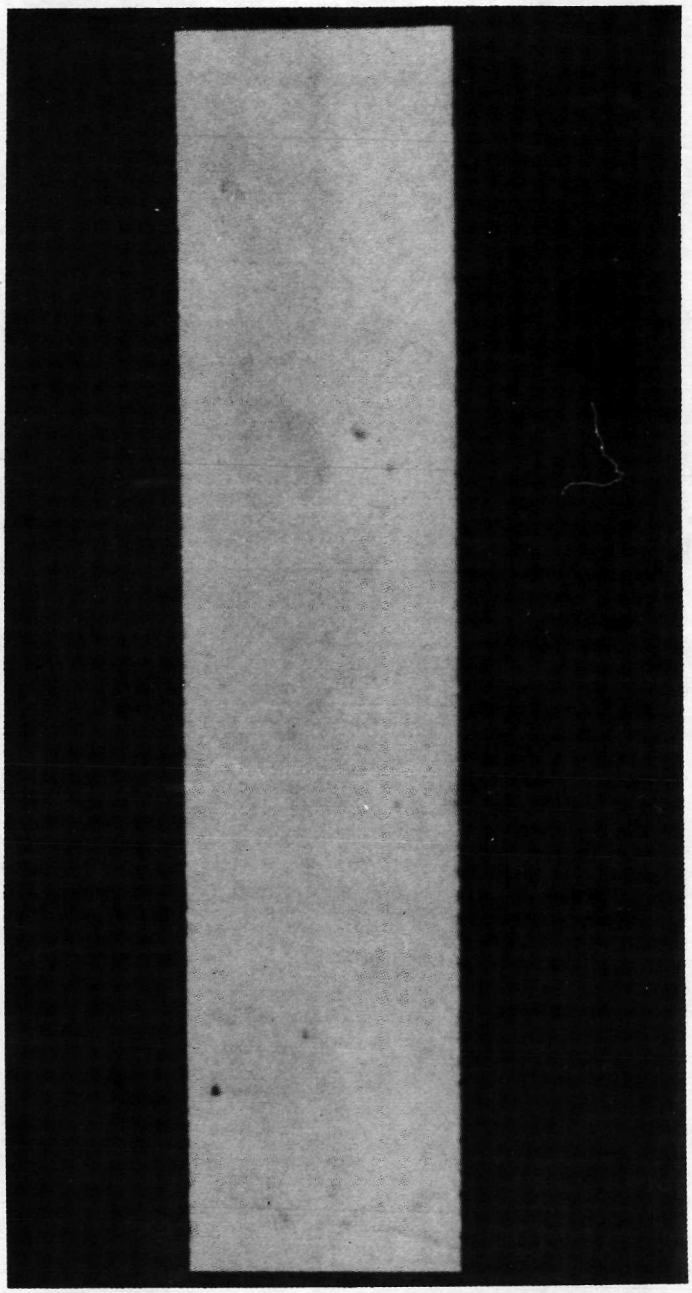


Figure 4.12 Radiographic Image of a Metal (II)

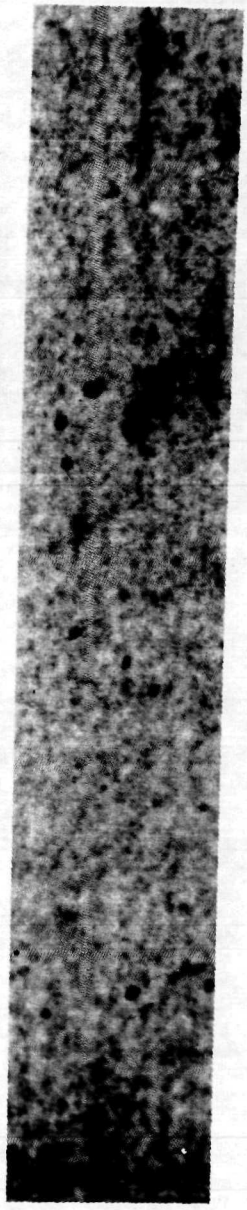


Figure 4.13 Linearly Rescaled Version of Figure 4.12

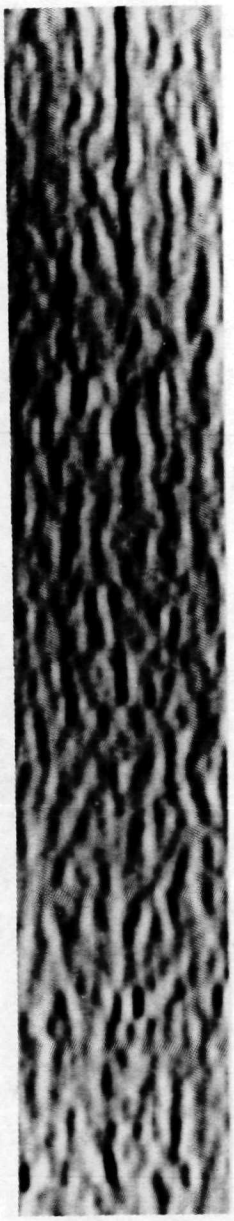


Figure 4.14 Result of Filtering Figure 4.13 with a Matched Filter to Enhance Dark Vertical Lines

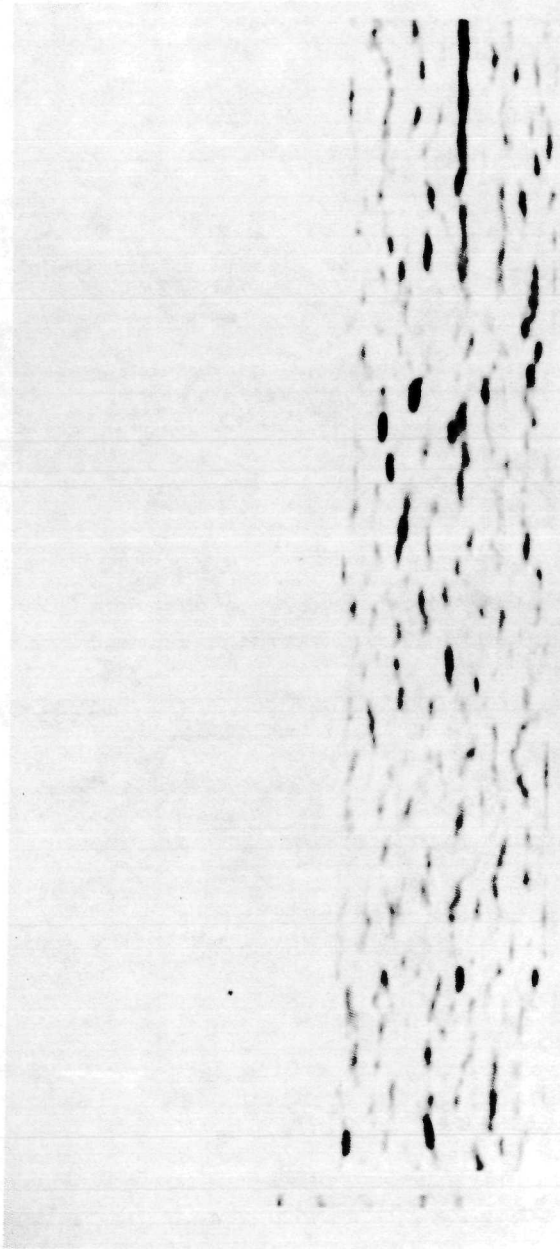


Figure 4.15 Result of Thresholding the Matched Filter Output in Figure 4.14 Before Rescaling

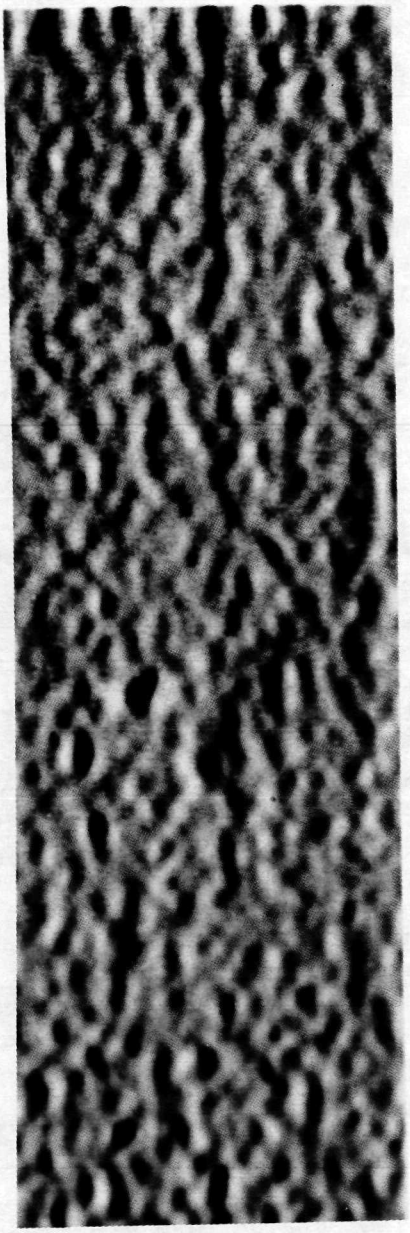


Figure 4.16 Result of Point-by-Point Addition of Densities in Figures 4.13 and 4.15

SECTION V CONCLUSIONS

The application of existing techniques of digital signal processing to radiographic image enhancement has been considered. A versatile set of programs has been developed to perform the image handling and image processing operations. These programs have been designed on the basis of a tape-drive-based IBM 7094.

The main image handling operations considered are: (i) converting the digitized data generated by a microdensitometer into a format suitable for use on the computer, (ii) automatic extraction of the data corresponding to the region of interest from a tape generated by the microdensitometer, (iii) transposition and 90^{o} rotations of large data arrays (which do not fit in core) (iv) translation of data arrays for registration (v) reduction of the dimensions of data arrays by integral factors and (vi) reconverting the data from computer format to a format suitable for "Photowriting" or display so that several frames of data can be centered and written on film or displayed on a screen.

Both the frequency domain and the spatial domain approaches have been considered in designing the image processing software. Programs have been developed using the Fast Fourier Transform for the design of filters with arbitrary continuous transfer functions to within a specified error, so that the number of filter weights is as small as possible. These programs are useful for designing digital filters which can enhance, attenuate or stop any desired ranges of spatial frequencies in the pictures. Fast implementation of the digital filters have been developed using fast convolution techniques and for some special cases of digital filters using recursive techniques. It has been found that the recursive implementation is much faster in the cases of interest in the present work, particularly for enhancing vertical or horizontal lines in radiographs. In these cases matched filters (template matching filters) have been implemented very fast using the recursive method (taking about 14 minutes for filtering a 1000 x 320 array).

Examples of test patterns have been used in this report to illustrate the techniques and the application of the techniques to radiographic images has been shown through several examples.

The choice of filter characteristics-transfer functions in the frequency domain, template sizes and certain thresholds on normalized cross correlation in the case of matched filtering - is still empirical. The way they are chosen at present is to filter a small area of the image with several filters and choose the filter which gives the best (visual) results. A knowledge of the statistical characteristics of noise in the radiographic imaging systems will help in the choice of the filter parameters. But the results obtained by the empirical approach are quite encouraging.

It is felt that further developmental effort should be in the areas of template matching and other particular-feature-oriented approaches. Cases in point are a fast approach for registration using sequential similarity detection [20] which can be applied to template matching and image pattern recognition and "Edge and Curve Detection" reported by Rosenfeld et al [21] (July 1972) which are used for the detection of "texture edges" in images.

PART II

SECTION I. INTRODUCTION

This part of the report presents the formal documentation of the image handling and processing software discussed in Part I. Section II presents the details of the programs for image handling operations such as automatic picture extraction, transposition and 90^{o} rotations of picture data, generation of reflections of picture data about the top and bottom edges and packing of data for multiple image display. Section III presents the details of the programs for image processing operations including design and implementation of filters in frequency domain, point operations on pictures such as rescaling and density manipulation and fast recursive implementation of a particular class of filters. Complete optimization of the programs has not yet been attempted and some of the programs can be improved considerably.

SECTION II. IMAGE HANDLING PROGRAMS

A1. Automatic Picture Extraction

1. NAME

Deck Name: APEXS

2. PURPOSE

To determine the region of interest in a digitized picture, extract it, unpack it and linearly rescale the data (as an option) and write it in unpacked format on a tape. Also, as an option, to override the determination of the region of interest and just extract, unpack and write a specified rectangular area of a digitized picture.

3. CALLING SEQUENCE

CALL APES (IA, IX, ILIM, IAMX, IAMN)

where all the calling arguments are integer work arrays to be dimensioned as described in the comments in the listing attached.

See also paragraph 7.3 below.

4. INPUT-OUTPUT

4.1 Input: The input tape should be on logical unit 8. (See note on buffer size for unit 8 in the comments in the listing.) The input tape is the tape produced by the microdensitometer and is assumed to contain integers between 0 and 63 packed as 6 pixels per computer word.

4.2 Output: The output of the program will be on logical unit 10, written in FORTRAN binary format, each record consisting of the record number followed by the data in unpacked format.

4.3 File Storage: None.

5. EXITS

When the program is used in the mode where the region of the picture to be extracted is to be determined, the program exits giving the message "NO RECORDS WERE EXTRACTED SINCE THE DENSITIES WERE ALL .GT. 62" if either there are no records in the input file with density numbers less than 63 or the number of successive occurrences of such records after the first such record has been found is less than or equal to the vertical margin MV1 (see paragraph 7.3 below).

While determining the region to be extracted, the program finds the first and last points to be extracted in each record. The initial column number is taken to be the right-most of the initial points of all the records to be extracted and final column number is taken to be the left-most of the final points of all the records to be extracted (see technique description in Part I, Section II-A, for details). If the initial column number exceeds the final column number (after margins are allowed), then the program exits, printing the message "NO RECORDS WERE EXTRACTED SINCE INITIAL COLUMN EXCEEDED FINAL COLUMN." In this case, the "frame skeleton details," i.e., the initial and final points in each record which are regarded by the program as belonging to the region of interest (after allowing specified margins) will be printed also.

In either of the above cases, no records will be written on unit 10. The messages will be printed both on-line and off-line. The frame skeleton details will be printed only off-line.

6. USAGE

The program is written in FORTRAN IV. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System Subroutines: SYSLOC, SKRBIN, BSFILE

7.2 Other Programs Called: UPVECT, UPVEC and BECOL (See Sections II.B and II-C.)

7.3 External Storage Used:

(1) COMMON/INAPE/HSL,SINT,MH1,MH2,MV1,MV2,ISCAL,IFLG

All parameters in this common block are inputs. The definitions of each parameter are given in comments in the listing.

(2) COMMON/OVRID/IRINIT,IRFIN,ICI,ICF,IFLG1

All parameters in this common block are inputs. The definitions of each parameter are given in comments in the listing.

Note: If the determination of the region of interest is to be overridden (i.e., IFLG1≠0), only HSL and SINT need be specified in the common block /INAPE/. If IFLG1=0, then no parameter other than IFLG1 need be specified in the common block /OVRID/.

(3) COMMON/OUAPE/NREC,NEL

Both the parameters are outputs giving the number of records and number of pixels per record which were actually extracted and written on unit 10.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 2176_8 locations of core.

8.2 Execution time: The execution time is highly dependent on the size of the image scanned and the size of the window being extracted. It also depends on the location of the window relative to the first record. The following figures are quoted as an example. In the test case attached, it can be seen that the time for determining the region of interest, unpacking and rescaling was about 7.5 minutes including tape rewinding at the end of the job.

8.3 I/O load: Under conditions not covered in paragraph 5 above, the program produces an output tape on logical unit 10. The values of the initial row, initial column, final row and final column extracted will be printed both on-line and off-line. The histogram of the densities in the region extracted will be printed both on-line and off-line. If scaling is requested (i.e., ISCAL≠0), the histograms of both the original and rescaled versions will be given in the off-line output. If IFLG≠0, then the "frame skeleton" details and the minimum and maximum value of the densities in each record in the region of interest will be printed.

8.4 Restrictions: None.

9. METHOD

See Part I, Section II-A, for a description of the method.

10. COMMENTS

The program might give the second message discussed under paragraph 5 due to skewness of the rectangular region of interest with respect to the scanning direction. In this case, it is suggested that the program be retried with either manual override (IFLG1≠0) or with increased vertical margins (MV1 and MV2).

11. LISTING

A listing of the program is attached at the end of Section II-A3.

12. TESTS

The program has been tested on many digitized data tapes which were generated by masking all but the desired rectangular regions in several radiographs. It has been found to work satisfactorily in all cases where most of the density numbers were less than 63. Typical values of margins to be used are 25 to 40 pixels, the required margins being larger for smaller scanning intervals. A typical output of a successful run is shown at the end of the listing.

A2. Unpacking a Specified Part of a Record

1. NAME

Deck name: UNPVEC

Entries: UPVECT and UPVEC

2. PURPOSE

To read a record of packed data and unpack a specified part of it (UPVECT) or to unpack a specified part of a given record (UPVEC).

3. CALLING SEQUENCE

CALL UPVECT (INITC, NEL, IX, IA, N, NERR)

or

CALL UPVEC (INITC, NEL, IX, IA, N)

IX is the integer work array into which the packed record is read.

IA is the integer output array into which the desired part of IX will be unpacked.

INITC = input integer giving the unpacked word number in IX which should go into IA(1). (See paragraph 9 below.)

NEL = input integer giving the number of (packed) words to be read into IX from the input tape (UPVECT) or the number of words in IX (UPVEC).

N = input integer giving the number of words desired in the output array IA.

NERR = output integer equal to 0 if the routine does not read an end of file and equal to 1 if it does.

4. INPUT-OUTPUT

4.1 Input: The input data for entry UPVECT is assumed to be on logical unit 8. The buffer size for unit 8 should be at least NEL. The tape is assumed to contain integers between 0 and 63 packed as 6 pixels per computer word. For entry UPVEC, the input is through the calling sequence only.

4.2 Output: The output is through the calling sequence only.

4.3 File storage: None.

5. EXITS

When the program UPVECT encounters an end of file mark, it sets all words in IA equal to 0, prints the message "EOF ENCOUNTERED BY UPVECT" both on-line and off-line and returns.

6. USAGE

The program is written in FORTRAN IV. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC, REDTPR

7.2 Other programs called: UNPINT (a program which unpacks a given word into an array of 6 integer words).

7.3 External storage used: None.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 354_8 core locations

8.2 Execution time: TBD

8.3 I/O load: None.

8.4 Restrictions: None.

9. METHOD

The program UPVECT reads NEL words of a record of data from the input tape into IX, using REDTPR. When unpacked, the array IX would thus yield 6 NEL words. It is desired to transfer N words out of these 6 NEL words into the array IA, starting from the INITCth word. The program determines the word number in IX corresponding to the INITCth unpacked word and unpacks only the number of words required to produce N unpacked words. Also, if (N + INITC - 1) exceeds the number of words NW read from tape (as determined by the routine REDTPR) the last few words of IA will be set to 0. The operation of UPVEC is the same except that the data is not read from the tape, but is already in core. Also, in UPVEC, NW is set equal to NEL first and then the unpacking proceeds as in the case of UPVECT.

10. COMMENTS

None.

11. LISTING

A listing of the program is attached at the end of Section A3.

12. TESTS

The program has been tested with known, packed test patterns and also in conjunction with the routine APES discussed in Section II-A1 and has been found to work satisfactorily.

A3. Determination of the Part of Interest in a Record

1. NAME

Deckname: BECLMN

Subroutine name: BECOL

2. PURPOSE

To find two points in a packed record such that all pixels before the first point and after the last point have densities equal to 63.

3. CALLING SEQUENCE

CALL BECOL (IX, N, I1, I2, K)

Where IX is an integer input vector consisting of N words, I1, I2 and K are output integers (see paragraph 9 below).

4. INPUT-OUTPUT

The input and output are through the calling sequence only.

5. EXITS

No abnormal exits.

6. USAGE

The program is written in FORTRAN IV. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC

7.2 Other programs called: UNPINT

7.3 External storage: None

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 260_8 core locations

8.2 Execution time: TBD

8.3 I/O load: None

8.4 Restrictions: None

9. METHOD

First of all, I1, I2 and K are set equal to 0. Next, I1 is determined so that

$$I1 = \text{Min} \left\{ I \mid 3 \leq I \leq N \text{ and } IX(I) \neq -377777777777_8 \right\}$$

That is, I1 is the index of the first word of IX, all of whose 36 bits are not "set". If such an I1 cannot be found, then the program returns with I1 = I2 = K = 0. If such an I1 is found, I2 is then determined such that

$$I2 = \text{Max} \left\{ I \mid 3 \leq I \leq N-3 \text{ and } IX(I) \neq -377777777777_8 \right\}$$

(Here, only $I \in [3, N-3]$ is considered to allow for scanner label words.)

Next, IX(I1) and IX(I2) are unpacked into 6 integer words each. Of the 6 integers resulting from IX(I1), let the J1th word be the first which is less than 63. Also, of the 6 integer words from IX(I2), let the J2th word be the last which is less than 63. Then, I1 and I2 are redefined as

$$I1 = (I1 - 1) * 6 + J1$$

and

$$I2 = (I2 - 1) * 6 + J2$$

and K is defined by

$$K = I2 - I1 + 1.$$

Thus, if IA denotes a 6N vector obtained by unpacking IX, then IA(I1) and IA(I2) are the first and last words of IA which are less than 63 and, hence, I1 and I2 mark the region of interest in the record. Also, K gives the length of the part of IA which is in the region of interest.

10. COMMENTS

None.

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

The program has been tested in conjunction with the routine APES (Section II-A1) and has been found to work satisfactorily.

MAIN - EFN SOURCE STATEMENT - IFN(S) - 08/01/72 PAGE 1

```
      DIMENSION IA(1500),IX(1500),ILIM(2,2500),IAMX(2500),IAMN(2500)
      COMMON/INAPE/HSL,SINT,MH1,MH2,MV1,MV2,ISCAL,IFLG
      COMMON/OUAPE/NREC,NEL
      COMMON/OVRID/IRINIT,IRFIN,ICI,ICF,IFLG1
      CALL STRIMR                                                    2
      CALL PET(0)                                                    4
      WRITE(6,100)                                                   5
      CALL APES(IA,IX,ILIM,IAMX,IAMN)                                6
      CALL PET(1)                                                    8
      REWIND 8                                                       9
      REWIND 10                                                      10
      REWIND 10                                                      11
      CALL PET(1)                                                    12
      IFLG1=1
      WRITE(6,100)                                                   14
100   FORMAT(1H1)
      REWIND 8                                                       15
      STOP
      END
```

BKDAT - EFN SOURCE STATEMENT - IFN(S) -

```
      BLOCK DATA
      COMMON/INAPE/HSL,SINT,MH1,MH2,MV1,MV2,ISCAL,IFLG
      COMMON/OVRID/IRINIT,IRFIN,IC1,ICF,IFLG1
      DATA IFLG1/0/
      DATA HSL,SINT,MH1,MH2,MV1,MV2,ISCAL,IFLG/15.,12.5,40,40,40,40,1,0/
      END
```

```
      SUBROUTINE APES(IA,IX,ILIM,IAMX,IAMN)
C     IA,IX SHD BE D*NED NSMPL=HSL*1000./SINT-5.
C     IAMX, IAMN SHD BE D*NED NREC=MAX NO. OF RECORDS EXPECTED. ILIM SHD
C     BE D*NED (2,NREC). IF IFLG1=1 IN /OVRID/ THEN IAMX,IAMN,ILIM ARE
C     NOT NEEDED.
C     THE BUFFER SIZE FOR UNIT 8 SHD BE AT LEAST NPSMPL=NSMPL/6+1.
C     AUTOMATIC PICTURE EXTRACTION AND SCALING. THIS PROGRAM ASSUMES THA
C     THE AREA OF THE PICTURE TO BE EXTRACTED HAS BEEN ISOLATED DURING
C     SCAN BY COVERING THE REST OF THE PICTURE BY A HIGH DENSITY MATERIA
C     (GIVING NUMBERS GT 62)
C     HSL=HORIZONTAL SCAN LENGTH IN MM AND SINT=SCANNING INTERVAL IN
C     MICRONS.
C     MH1 AND MH2 ARE HORIZONTAL MARGINS(LEFT AND RIGHT). MV1,MV2 ARE
C     VERTICAL MARGINS(TOP, BOTTOM). THE MARGINS ARE IN PIXELS.
C     NREC,NEL ARE OUTPUTS GIVING THE NUMBER OF RECORDS AND NUMBER OF
C     PIXELS PER RECORD EXTRACTED.
C     ISCAL SHD BE GIVEN AS 0 IF SCALING IS NOT DESIRED. IELG=0 IF A DET
C     DETAILED PRINTOUT OF PICTURE FRAME SKELETON AND MAX AND MIN OF EA
C     EACH RECORD ARE NOT DESIRED.
C     COMMON BSOCK /OVRID/ PROVIDES THE CAPABILITY TO BYPASS THE DETER
C     MINATION OF THE AREA TO BE EXTRACTED AND TO SPECIFY IT EXTERNALLY.
C     IN THIS CASE IFLG1 SHD BE GIVEN AS NONZERO AND IRINIT, IRFIN, ICI,
C     ICF SHD BE GIVEN AS INITIAL ROW, FINAL ROW, INITIAL COLUMN, FINAL
C     COLUMN TO BE EXTRACTED (NO SCALING).
      COMMON /INAPE/HSL,SINT,MH1,MH2,MV1,MV2,ISCAL,IFLG
      COMMON /OUAPE/NREC,NEL
      COMMON/OVRID/IRINIT,IRFIN,ICI,ICF,IFLG1
      DIMENSION IA(1),IX(1),ILIM(2,1),IAMX(1),IAMN(1)
      DIMENSION IH1(64),IH2(64)
      DIMENSION ISCT(64)
      LOGICAL L
      L=ISCAL.NE.0.AND.IFLG1.EQ.0
      DO 120 I=1,64
      IH1(I)=0
120   IH2(I)=0
      NSMPL=HSL*1000./SINT-5.
      NPSMPL=NSMPL/6+1
      IAMIN=100
      IAMAX=0
      IF(IFLG1.NE.0) GO TO 210
      NREC1=0
      NREC=0
      IRINIT=0
      IRFIN=0
20    CONTINUE
      CALL REDTPR(8,2,NERR,NW,NPSMPL,IX)                    23
      IF(NERR.NE.2.AND.NERR.NE.5)GO TO 15
      NERR=1
      GO TO 100
15    NERR=0
      NREC=NREC+1
      NW=NW-2
      CALL BECOL(IX,NW,I1,I2,K)                             33
      ILIM(1,NREC)=I1
      ILIM(2,NREC)=I2
```

```
C     ALLOW HORIZONTAL MARGINS.
      ILIM(1,NREC)=ILIM(1,NREC)+MH1
      ILIM(2,NREC)=ILIM(2,NREC)-MH2
C     NREC1 COUNTS THE NUMBER OF RECORDS WHICH CONTAIN POINTS OF DENSITY
C     .LE. 62. WHEN NREC1 REACHES MV1+1, THE FIRST RECORD TO BE EXTRACTE
C     IS OBTAINED. IF NREC1 IS NOT 0 AND K IS 0, IT MEANS THAT THE END
C     OF THE REGION TO BE EXTRACTED IS REACHED.
      IF(K.NE.0) NREC1=NREC1+1
      IF(NREC1.EQ.MV1+1) IRINIT=NREC
      IF(NREC1.NE.0.AND.K.EQ.0) GO TO 100
      IF(ISCAL.EQ.0) GO TO 45
      I2=ILIM(2,NREC)
      I1=ILIM(1,NREC)
      K=I2-I1+1
      IF(K.LE.0)GO TO 45
      CALL UPVEC(I1,NW,IX,IA,K)                                    59
      IAMX(NREC)=0
      IAMN(NREC)=100
      DO 40 I=1,K
      IF(IAMX(NREC).LT.IA(I))IAMX(NREC)=IA(I)
      IF(IAMN(NREC).GT.IA(I))IAMN(NREC)=IA(I)
40    CONTINUE
45    CONTINUE
      IF(NREC).EQ.0.OR.K.NE.0) GO TO 20
C     STATEMENT 100 CAN BE REACHED WITH NERR=1 OR NERR=0. IF NERR=1 NREC
C     IS THE NUMBER OF RECORDS IN THE FILE. IF NERR=0 NREC=THE FIRST
C     RECORD AT THE END OF THE REGION OF INTEREST. THE FINAL RECORD TO
C     BE EXTRACTED IS DETERMINED BY ALLOWING THE SPECIFIED VERTICAL MARG
100   IRFIN=NREC-1-MV2
C     BSFILE STATEMENTS TAKE THE TAPE UNIT TO THE BEGINNING OF THE FILE
C     UNDER CONSIDERATION.
      IF(NERR.EQ.0) CALL BSFILE(8,1)                               88
      IF(NERR.EQ.1) CALL BSFILE(8,2)                               91
      IF(IFLG.EQ.0) GO TO 80
      WRITE(6,1000)(ILIM(1,J),J=1,NREC)                            95
      WRITE(6,1100)                                                102
      WRITE(6,1000)(ILIM(2,J),J=1,NREC)                            103
      WRITE(6,1100)                                                110
1000  FORMAT(2X,2015)
1100  FORMAT(/)
      IF(ISCAL.EQ.0) GO TO 140
      WRITE(6,1000)(IAMX(J),J=1,NREC)                              113
      WRITE(6,1100)                                                120
      WRITE(6,1000)(IAMN(J),J=1,NREC)                              121
      WRITE(6,1100)                                                128
80    CONTINUE
C     THE MAX AND MIN OF DENSITIES ARE DETERMINED OVER THE LIGHTER AREA
C     OF THE PICTURE ISOLATED DURING THE SCAN MINUS THE SPECIFIED MARGIN
C     THESE MAX AND MIN COULD BE GREATER AND LESS RESPECTIVELY THAN THOS
C     THOSE FOR THE AREA WHICH WILL BE FINALLY EXTRACTED.
      DO 110 IR=IRINIT,IRFIN
      IF(IAMAX.LT.IAMX(IR))IAMAX=IAMX(IR)
110   IF(IAMIN.GT.IAMN(IR))IAMIN=IAMN(IR)
C     A SCALING TABLE IS COMPUTED ON THE BASIS OF THE ABOVE AMAX AND
C     AMIN.
      RANGE=IAMAX-IAMIN
```

```
      DO 180 I=1,64
      IF(RANGE.EQ.0.)ISCT(I)=IAMAX
      IF(RANGE.NE.0.)ISCT(I)=63.*FLOAT(I-1-IAMIN)/RANGE+.5
180   CONTINUE
140   CONTINUE
      IF(IRINIT.EQ.0) GO TO 30
210   CONTINUE
      IR1=IRINIT-1
      IF(IR1.EQ.0) GO TO 190
      CALL SKRBIN(8,IR1,IER)                                          165
190   CONTINUE
      IF(IFLG1.NE.0) GO TO 220
C     THE TAPE UNIT 8 IS READY TO READ IRINIT TH RECORD.
      ICI=0
      ICF=100000
      DO 50 IR=IRINIT,IRFIN
      IF(ICI.LT.ILIM(1,IR)) ICI=ILIM(1,IR)
50    IF(ICF.GT.ILIM(2,IR)) ICF=ILIM(2,IR)
C     THE LARGEST ILIM(1,IR) AND THE SMALLEST ILIM(2,IR) ARE FOUND TO
C     DETERMINE THE FIRST AND LAST COLUMNS TO BE EXTRACTED.
220   CONTINUE
      PRINT 600,IRINIT,IRFIN,ICI,ICF                                  186
      WRITE(6,600) IRINIT,IRFIN,ICI,ICF                               187
600   FORMAT(10X,7HIRINIT=,I4,7H IREIN=,I4,5H ICI=,I4,5H ICF=,I4)
      ICT=ICF-ICI+1
      IF(ICT.LE.0) GO TO 35
      DO 60 IR=IRINIT,IRFIN
      CALL UPVECT(ICI,NPSMPL,IX,IA,ICT,NERR)                          195
      DO 70 I=1,ICT
      JJ=IA(I)+1
      IH1(JJ)=IH1(JJ)+1
      IF(L)GO TO 230
      IX(I)=IA(I)
      GO TO 70
230   CONTINUE
      IX(I)=ISCT(JJ)
      JJ=IX(I)+1
      IH2(JJ)=IH2(JJ)+1
70    CONTINUE
150   CONTINUE
      JR=IR-IRINIT+1
      WRITE(10) JR,(IX(I),I=1,ICT)                                    219
60    CONTINUE
      IF(L)PRINT 200,JR,ICT,IRINIT,ICI                                227
      IF(.NOT.L)PRINT 201,JR,ICT,IRINIT,ICI                           229
      IF(L) WRITE(6,200)JR,ICT,IRINIT,ICI                             231
      IF(.NOT.L) WRITE(6,201)JR,ICT,IRINIT,ICI                        233
200   FORMAT(10X,75HTHE OUTPUT ON A6 IS THE RESCALED VERSION OF THE REGI
     *ON OF INTEREST WHICH IS,I5,1H*,I4,21H PART STARTING FROM (,I4,1H,,
     214,2H).)
201   FORMAT(10X,51HTHE OUTPUT ON A6 IS THE REGION OF INTEREST WHICH IS,
     115,1H*,I4,21H PART STARTING FROM (,I4,1H,,I4,2H).)
      IF(L) PRINT 300,IAMIN,IAMAX                                     235
      IF(L)WRITE(6,300)IAMIN,IAMAX                                    237
300   FORMAT(10X,13HMIN. DENSITY=,I2,14H MAX. DENSITY=,I2)
      NREC=JR
```

```
      NEL=ICT
      IF(L)WRITE(6,1200)                                                241
      IF(.NOT.L)WRITE(6,1201)                                           243
1200  FORMAT(//,10X,8HDENSITY ,9X,8HORIGINAL,10X,8HRESCALED,/)
1201  FORMAT(//,10X,8HDENSITY ,9X,9HFREQUENCY,/)
      DO 130 I=1,64
      J=I-1
      IF(L)WRITE(6,1300)J,IH1(I),IH2(I)                                 249
      IF(.NOT.L)WRITE(6,1300)J,IH1(I)                                   253
      PRINT 1300,J,IH1(I)                                               255
130   CONTINUE
1300  FORMAT(15X,I2,10X,I8,10X,I8)
      RETURN
30    CONTINUE
      NREC=0
      NEL=0
      WRITE(6,700)                                                      262
      PRINT 700                                                         263
700   FORMAT(/,10X,62HNO RECORDS WERE EXTRACTED SINCE THE DENSITIES WERE
     1 ALL .GT. 62,/)
      RETURN
35    CONTINUE
      WRITE(6,701)                                                      264
701   FORMAT(/,10X,69HNO RECORDS WERE EXTRACTED SINCE INITIAL COLUMN EXC
      EEDED FINAL COLUMN.)
      IF(IFLG.NE.0) GO TO 36
702   FORMAT(10X,41HFRAME SKELETON DETAILS ARE PRINTED BELOW.,/)
      WRITE(6,702)                                                      267
      WRITE(6,1000)(ILIM(1,J),J=1,NREC)                                 268
      WRITE(6,1100)                                                     275
      WRITE(6,1000)(ILIM(2,J),J=1,NREC)                                 276
      WRITE(6,1100)                                                     283
36    CONTINUE
      NREC=0
      NEL=0
      RETURN
      END
```

```
      SUBROUTINE UPVECT(INITC,NEL,IX,IA,N,NERR)
C     SUBROUTINE TO UNPACK A VECTOR IX WITH NEL PACKED WORDS INTO IA, AN
C     N VECTOR STARTING FROM UNPACKED WORD NUMBER INITC. (INITC.GE.1).
C     BUFFER SIZE FOR UNIT 8 SHD BE AT LEAST NEL.
C     IF EOF IS ENCOUNTERED THE PROGRAM SETS NERR=1. OTHERWSIE NERR=0
C     WHEN IA EXCEEDS THE RANGE OF IX READ FROM TAPE THE LAST FEW
C     ELEMENTS OF IA WILL BE SET TO ZERO.
      DIMENSION IX(NEL),IA(N),IW(6)
      CALL REDTPR(8,2,NERR,NW,NEL,IX)                                    2
      IF(NERR.EQ.2.OR.NERR.EQ.5)GO TO 20
      NERR=0
      GO TO 50
      ENTRY UPVEC(INITC,NEL,IX,IA,N)
      NW=NEL
50    CONTINUE
      IC1=(INITC-1)/6
      ICR=INITC-1-IC1*6
      N1=(N-(6-ICR))/6
      NR=N-(6-ICR)-N1*6
      IF(NR.EQ.0)N2=N1+1
      IF(NR.NE.0)N2=N1+2
      DO 10 I=1,N
10    IA(I)=0
      DO 30 I=1,N2
      J=I+IC1
      IF(J.GT.NW)RETURN
      CALL UNPINT(IX(J),IW)                                              31
      J1=6*(I-1)+1-ICR
      J2=J1+5
      IF(I.EQ.1)J1=1
      IF(I.EQ.N2)J2=N
      IF(J2.LT.J1)GO TO 30
      DO 40 JJ=J1,J2
      JJJ=JJ-J1+1
      IF(I.EQ.1)JJJ=JJ-J2+6
40    IA(JJ)=IW(JJJ)
30    CONTINUE
      RETURN
20    NERR=1
      WRITE(6,100)                                                       56
      PRINT 100                                                          57
100   FORMAT(10X,25HEOF ENCOUNTERED BY UPVECT)
      RETURN
      END
```

```
      SUBROUTINE BECOL(IX,N,I1,I2,K)
C     TO FIND POINTS IN PACKED RECORD IX SUCH THAT IA(I1).LT.63, IA(I2)
C     .LT.63 AND IA(I).EQ.63 FOR 1.LE.I.LT.I1 AND I2.LT.I.LE.N( WHERE IA
C     IS THE UNPACKED VERSION OF IX( IA HAS 6 TIMES AS MANY WORDS AS IX)
C     NOTE THAT ALL OF IX NEED NOT BE UNPACKED TO FIND I1 AND I2.
C     IF IA(I).EQ.63 FOR ALL I, THEN I1=I2=K=0 AT THE END OF CALL. OTHER
C     WISE K=I2-I1+1 (THE FIRST TWO WORDS OF IX WILL BE SKIPPED TO ALLOW
      DIMENSION IX(N),IW(6)
      DATA ISET/O-377777777777/
C     FOR LABEL WORDS OF THE SCANNER).
      K=0
      I1=0
      I2=0
      DO 10 I=3,N
      IF(IX(I).EQ.ISET)GO TO 10
      I1=I
      GO TO 20
10    CONTINUE
      RETURN
20    DO 30 I=1,N
      J=N-I+1
      IF(J.LE.2.OR.IX(J).EQ.ISET)GO TO 30
      I2=J
      GO TO 40
30    CONTINUE
40    CALL UNPINT(IX(I1),IW)
      DO 50 I=1,6
      IF(IW(I).EQ.63)GO TO 50
      K=I
      GO TO 60
50    CONTINUE
60    I1=6*(I1-1)+K
      CALL UNPINT(IX(I2),IW)
      DO 70 I=1,6
      J=6-I+1
      IF(IW(J).EQ.63)GO TO 70
      K=J
      GO TO 80
70    CONTINUE
80    I2=6*(I2-1)+K
      K=I2-I1+1
      RETURN
      END
```

IRINIT= 440 IRFIN=2146 ICI= 412 ICF= 777
THE OUTPUT ON 46 IS THE RESCALED VERSION OF THE REGION OF INTEREST WHICH IS 1707* 366 PART STARTING FROM (440, 412).
MIN. DENSITY=32 MAX. DENSITY=45

DENSITY	ORIGINAL	RESCALED
0	0	97
1	0	0
2	0	0
3	0	0
4	0	0
5	0	8707
6	0	0
7	0	0
8	0	0
9	0	0
10	0	124333
11	0	0
12	0	0
13	0	0
14	0	0
15	0	317515
16	0	0
17	0	0
18	0	0
19	0	155846
20	0	0
21	0	0
22	0	0
23	0	0
24	0	17127
25	0	0
26	0	0
27	0	0
28	0	0
29	0	808
30	0	0
31	0	0
32	97	0
33	8707	0
34	124333	172
35	317515	0
36	155846	0
37	17127	0
38	808	0
39	172	69
40	69	0
41	39	0
42	18	0
43	16	0
44	10	39
45	5	0
46	0	0
47	0	0
48	0	18
49	0	0
50	0	0
51	0	0
52	0	0
53	0	16

54	0	0
55	0	0
	0	0
58	0	10
59	0	0
60	0	0
61	0	0
62	0	0
63	0	5

TIME ELAPSED SINCE LAST PRINTING OF TIME= 0.387E 03SEC. TOTAL TIME ELAPSED= 0.387E 03SEC.
TIME ELAPSED SINCE LAST PRINTING OF TIME= 0.615E 02SEC. TOTAL TIME ELAPSED= 0.449E 03SEC.

B. Transposition and 90° Rotations of Image Data

1. NAME

Deck name: MTROT

Entries: MTRANP and MROT

2. PURPOSE

To generate transposition or 90° rotations (clockwise or counterclockwise) of an MXN data array stored on tape as M records of N words each (N + 1 words, in fact, the first word being the record number), when $M \leq 1024$ and $N \leq 1024$.

3. CALLING SEQUENCE

For transposing:

CALL MTRANP (NR, NC, IX, NTAPI, NTAPO)

For 90° rotation:

CALL MROT (NR, NC, IX, NTAPI, NTAPO, IROT)

where

NR = Number of rows in the input array.

NC = Number of columns in the input array.

IX = Work array which can be real or integer.

NTAPI = Input tape unit.

NTAPO = Output tape unit.

IROT = +1 for counterclockwise rotation and -1 for clockwise rotation.

4. INPUT-OUTPUT

4.1 Input: The input array should be on logical unit NTAPI. The records should be in FORTRAN binary format, each record containing the record number followed by NC words.

4.2 Output: The output array will be written on unit NTAPI in FORTRAN binary format, each record continuing the record number and followed by NR words of data.

4.3 File Storage: The number of scratch tapes used by the program depends on the size of the matrix to be transposed. In fact,

$$n = \text{Number of scratch tapes} = \text{Max} \left\{ \left[(NR-1)/128 \right] , \left[(NC-1)/128 \right] \right\} + 1$$

The first n of the tape units 2, 3, 4, 8, 10, 11, 12, 13 will be needed by the program. The mnemonics (B3, A4, ...) for the tape units needed will be printed on-line and the program will pause. The units requested should be provided before restarting the program.

5. EXITS

No non-standard exits.

6. USAGE

The program will run on any computer system with FORTRAN IV or a higher level of FORTRAN. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC, BSFILE

7.2 Other programs called: RQSTWT, RWNDWT, FLIPV, ASMBL (see listings attached)

7.3 External storage used: None

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 40770_8 locations of core

8.2 Execution time: The execution time depends on the size of the array to be transposed or rotated. Most of the time involved is for input/output operations and the time depends mainly on the minimum number of 128 x 128 partitions which can cover the given array. The time taken for some of the arrays which were transposed and rotated using this program are shown below.

Array Size	Time (Sec.) Transposition	Time (Sec.) 90^o Rotation
100 x 266	80.4	60.9
250 x 900	296	309
850 x 900	1020	1050

9. METHOD

See Section II-B of Part I.

10. COMMENTS

None.

11. LISTING

A listing of this program and the programs called by this program (listed in paragraph 7.2) are attached at the end of this section. The comments in the listings are sufficient to describe the operation of these subroutines.

12. RESTRICTIONS

The array sizes which can be handled are restricted only by the number of work tapes available.

```
      SUBROUTINE MTRANP(NR,NC,IX,NTAPI,NTAPO)
      INTEGER TAPE(8),TP
      DIMENSION IX(1),APT(128,128),NREST(2)
      IROT=0
      GO TO 70
      ENTRY MROT(NR,NC,IX,NTAPI,NTAPO,IRTN)
C     IRTN=+1 FOR COUNTERCLOCKWISE ROTATION BY 90 DEG. AND IRTN=-1 FOR
C     CLOCKWISE ROTATION BY 90 DEG.
      IROT=IRTN
70    CONTINUE
C     DETERMINE NUMBER OF TAPE UNITS NEEDED
      N=(NC-1)/128+1
      M=(NR-1)/128+1
      NTAP=MAKO(M,N)
      NREST(1)=NTAPI
      NREST(2)=NTAPO
      CALL RQSTMT(NTAP,TAPE,2,NREST)
C
C     WRITE A1,...,AN ON TAPE UNITS 1,...,N WHERE A1,...,A(N-1) ARE
C     (NR*128) PARTITIONS OF THE GIVEN MATRIX  A AND AN IS APPENDED WITH
C     ZERO COLUMNS IF NEEDED TO MAKE IT (NR*128). ALSO, THE NUMBER OF
C     ROWS IS MADE A MULTIPLE OF 128 BY APPENDING ZERO ROWS.
C                                                                       8
      NRP=M*128
      DO 10 I=1,NRP
      IF(I.GT.NR) GO TO 14
      READ(NTAPI,IR,(IX(J),J=1,NC)                                     16
      IF(IROT.EQ.1)CALL FLIPV(IX,IX,NC)
C                                                                      23
14    CONTINUE
      J2=0
      DO 10 K=1,N
      TP=TAPE(K)
      J1=J2+1
      J2=J2+128
      WRITE(TP)(IX(J),J=J1,J2)                                         32
10    CONTINUE
      CALL RWNDMT(TAPE,N)
C
C      READ 128*128 SUBMATRICES AIJ OF A1,...AN AND WRITE AI1T,AI2T,...
C     AMAT ON TAPE 14.
C                                                                      41
      DO 20 I=1,M
      DO 20 J=1,N
      TP=TAPE(J)
      DO 17 IR=1,128
17    READ (TP) (APT(IR,IC),IC=1,128)                                  52
      WRITE(NTAPO)((APT(IR,IC),IR=1,128),IC=1,128)                     57
20    CONTINUE
      END FILE NTAPO                                                   69
      CALL BSFILE(NTAPO,2)                                             70
      CALL RWNDMT(TAPE,N)
C
C     READ AI1T,AI2T,...,AMAT ONE BY ONE INTO ARRAY APT AND WRITE AIJT
C     FOR J=1,...,N ON TAPE I, I=1,...,M.
```

MTROT - EFN SOURCE STATEMENT - IFN(S) -

```
C                                                                       72
      DO 30 I=1,M
      TP=TAPE(I)
      DO 30 J=1,N
      READ(NTAPO)((APT(IR,IC),IR=1,128),IC=1,128)                       80
      DO 30 IR=1,128
      WRITE(TP) (APT(IC,IR),IC=1,128)                                   89
30    CONTINUE
      CALL BSFILE(NTAPO,1)                                              99
       CALL RWNDWT(TAPE,M)
C
C     READ ONE RECORD EACH FROM THE M TAPES AND FORM A ROW OF THE TRANS-
C     POSED MATRIX AND WRITE ON TAPE 14.
C                                                                       101
      DO 60 JREC=1,NC
      CALL ASMBL(IX,128,TAPE,M)                                         106
      IF(IROT.EQ.-1)CALL FLIPV(IX,IX,NR)                                109
      WRITE(NTAPO) JREC,(IX(I),I=1,NR)                                  110
60    CONTINUE
      CALL RWNDWT(TAPE,M)                                               118
      RETURN
      END
```

```
      SUBROUTINE RQSTWT(NWTP,NTPW,NRES,NREST)
C     THIS SUBROUTINE REQUESTS THE OPERATOR TO PROVIDE THE FIRST NWTP
C     WORK TAPES FROM THE ARRAY NTAPW DIFFERENT FROM THE NRES RESERVED
C     TAPES SPECIFIED BY THE ARRAY NREST. THE TAPE UNITS REQUESTED WILL
C     BE STORED IN THE ARRAY NTPW.
      DIMENSION NTPW(NWTP),NREST(NRES),NTAPW(10),TAPEU(10),RTAPU(10)
      DATA NTAPW/2,3,4,8,10,11,12,13,14,15/
      DATA TAPEU/2H03,2HA4,2HB4,2HA5,2HA6,2HB6,2HA7,2HB7,2HA9,2HB9/
      IF(NWTP.EQ.0)RETURN
      PRINT 100                                                    3
  100 FORMAT(//,10X,32HPROVIDE THE FOLLOWING WORK TAPES)
      J=0
      DO 10 I=1,NWTP
   30 J=J+1
      DO 20 K=1,NRES
   20 IF(NTAPW(J).EQ.NREST(K)) GO TO 30
      NTPW(I)=NTAPW(J)
   10 RTAPU(I)=TAPEU(J)
      PRINT 200,(RTAPU(I),I=1,NWTP)                                24
  200 FORMAT(10X,10A5,10(/))
      PAUSE
      RETURN
      END
```

RWWT - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE RWNDWT(TAPES,NT)
C     THIS SUBROUTINE REWINDS NT WORK TAPES.
      INTEGER TAPES(NT)
      IF(NT.EQ.0)RETURN
      DO 10 IT=1,NT
      NTP=TAPES(IT)
10    REWIND NTP                                                    8
      RETURN
      END
```

FLPV - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE FLIPV(X,Y,N)
      DIMENSION X(N),Y(N)
C     TO -FLIP- ELEMENTS OF AN N VECTOR X AND PUT THEM IN Y SUCH THAT
C     Y(I)=X(N-I+1) FOR I=1,...,N. X AND Y MAY BE THE SAME.
      IF(N.EQ.0)RETURN
      N1=N/2
      DO 10 I=1,N1
      I1=N-I+1
      W=X(I1)
      Y(I1)=X(I)
10    Y(I)=W
      RETURN
      END
```

ASKEWT - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE ASMBL(X,NEL,TAPES,NT)
C     TO ASSEMBLE INTP ARRAY X, NEL ELEMENTS EACH FROM EACH OF NT WORK T
C     TAPES. X SHO BE D-NEC AT LEAST NT*NEL.
      DIMENSION X(1)
      INTEGER TAPES(NT)
      IF(NT.EQ.0) RETURN
      DO 10 IT=1,NT
      NEL1=1+(IT-1)*NEL
      NEL2=NEL1+NEL-1
      NTP=TAPES(IT)
10    READ(NTP)(X(IEL),IEL=NEL1,NEL2)                                    10
      RETURN
      END
```

C. Reflection of a Picture in its Horizontal Edges

1. NAME

Deck name: RFLCT

Subroutine name: REFLC

2. PURPOSE

Given input picture data on a tape, to generate a file containing first the reflection of a given number n of data records in the first record, next the data records themselves and finally the reflection of n records of data in the last record.

3. CALLING SEQUENCE

CALL REFLC (W, NR1, NEL)

where W is a work array which can be integer or real and NR1 and NEL are input integers. The array W is dimensioned (NEL, NR1). (See the comments in the listing attached at the end of this section.)

4. INPUT-OUTPUT

4.1 Input: Input should be on logical unit NTAPI specified in the common block /INREFL/. (See paragraph 7.3 below.)

4.2 Output: Output will be on logical unit NTAPO specified in the common block /INREFL/.

4.3 File storage: The number of work tapes needed is given by

NWTP = (NRECR - 1)/NR1

where NRECR is the number of records to be reflected in the top (and bottom) edge of the picture. The work tapes needed will be requested by an on-line message to the operator followed by a pause.

5. EXITS

If the number of work tapes needed exceeds 8, then the program will print the message "ARRAY TOO LARGE TO HANDLE" and return.

6. USAGE

This program is written in FORTRAN IV and presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC, BSFILE

7.2 Other programs called: RQSTWT (See listing attached at the end of Section II-B.)

7.3 External storage used: The common block

COMMON/INREFL/NREC,NRECR,NTAPI,NTAPO

is used to provide the inputs. NREC is the number of input records. NRECR is the number of records to be reflected. NTAPI and NTAPO are respectively the input and output logic units.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 1633_8 locations of core

8.2 Execution time: The execution time is highly dependent on NREC, NR1, NEL and NRECR. The value of NR1 should be taken as large as possible so that the number of work tapes needed is minimized. The time taken was approximately 107 seconds for a test case where NREC = 1024, NR1 = 25, NEL = 333 and NRECR = 49. For a case with NREC = 1707, NR1 = 43, NEL = 366 and NRECR = 80, the program took approximately 164 seconds.

8.3 I/O load: None

8.4 Restrictions: The size of the region which can be reflected (i.e., NRECR*NEL) is limited by the number of work tapes available and the core size. In the present setup where a maximum of 8 work tapes is available, NRECR*NEL will be limited to approximately 128000.

9. METHOD

See Section II-C of Part I.

10. COMMENTS

None.

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

The program has been tested using a test pattern and also several picture data. Printouts of records and also images generated after packing the output indicate satisfactory operation of the program.

```
      SUBROUTINE REFLC(W,NR1,NEL)
      COMMON /INREFL/ NREC,NRECR,NTAPI,NTAPO
C     GIVEN A PICTURE HAVING NREC RECORDS, TO PRODUCE A PICTURE HAVING
C     NREC+2*NRECR RECORDS WHERE THE FIRST NRECR RECORDS ARE THE RECORDS
C     NRECR+1,...,2 OF THE ORIGINAL PICTURE, THE NEXT NREC RECORDS ARE
C     THE ORIGINAL PICTURE ITSELF AND THE LAST NRECR RECORDS ARE THE
C     RECORDS NREC-1,...,NREC-NRECR OF THE ORIGINAL PICTURE. OBVIOUSLY
C     NREC SHO BE .GT. NRECR.
C     NR1 IS AN ARBITRARY INTEGER SUCH THAT THE ARRAY W FITS IN CORE.
C     CHOICE OF NR1 GOVERNS THE NUMBER OF WORK TAPES NEEDED.
C     NOTE THAT IF NRECR IS AN INTEGRAL MULTIPLE OF NR1, THEN THE LAST
C     NRECR RECORDS OF OUTPUT WILL BE NREC,NREC-2,NREC-3,...,NREC-NRECR
C     TH RECORDS OF THE INPUT PICTURE.
      DIMENSION W(NEL,NR1)
      DIMENSION NTPW(10),NREST(2)
C     FIND THE NUMBER OF WORK TAPES NEEDED.
C
      NWTP=(NRECR-1)/NR1
      NR2=NRECR-NWTP*NR1
      READ(NTAPI) IR,((W(IEL,1),IEL=1,NEL)                              3
      IF(NWTP.EQ.0)GO TO 10
      IF(NWTP.GT.8)GO TO 20
      NREST(1)=NTAPI
      NREST(2)=NTAPO
      CALL RQSTWT(NWTP,NTPW,2,NREST)
C     LOOP 50 WRITES THE ITH SET OF NR1 RECORDS ON THE ITH WORK TAPE.
C                                                                       17
      DO 50 I=1,NWTP
      NTP=NTPW(I)
      DO 60 IREC=1,NR1
   60 READ(NTAPI)IR,(W(IEL,IREC),IEL=1,NEL)                             26
   70 WRITE(NTP)((W(IEL,IREC),IEL=1,NEL),IREC=1,NR1)                    33
      REWIND NTP                                                        41
   50 CONTINUE
C     LOOPS 90 AND 80 WRITE THE RECORDS STARTING FROM NWTP*NR1+2 TO
C     NRECR+1 ON NTAPO IN REVERSE ORDER.
C
   10 CONTINUE
      DO 80 IREC=1,NR2
   80 READ(NTAPI) IR,(W(IEL,IREC),IEL=1,NEL)                            49
      JREC=0
      DO 90 IREC=1,NR2
      IR=NR1-IREC+1
      JREC=JREC+1
   90 WRITE(NTAPO) JREC,(W(IEL,IR),IEL=1,NEL)                           62
      CALL BSFILE(NTAPI,1)                                              69
      IF(NWTP.EQ.0) GO TO 100
C     LOOP 110 WRITES THE RECORDS FROM WORK TAPES ON THE OUTPUT TAPE.
C     FOR I, THE RECORDS FROM (NWTP+1-I)TH TAPE WILL BE WRITTEN.
C
      DO 110 I=1,NWTP
      J=NWTP+1-I
      NTP=NTPW(J)
      READ(NTP)((W(IEL,IREC),IEL=1,NEL),IREC=1,NR1)                     79
      DO 120 IREC=1,NR1
```

```
RFLCT      - EFN  SOURCE STATEMENT  - IFN(S) -

      IR=NR1+1-IREC
      JREC=JREC+1
120   WRITE(NTAPO) JREC,(W(IEL,IR),IEL=1,NEL)                          90
110   REWIND NTP
C     NOW LOOP 130 WRITES THE FIRST NREC-(NRECR+1) RECORDS FROM THE
C     INPUT TAPE ONTO THE OUTPUT TAPE.
C                                                                      96
100   CONTINUE
      NREC1=NREC-NRECR-1
      IF(NREC1.EQ.0) GO TO 220
      DO 130 IREC=1,NREC1
      READ(NTAPI)IR,(W(IEL,1),IEL=1,NEL)                              106
      JREC=JREC+1
130   WRITE(NTAPO) JREC,(W(IEL,1),IEL=1,NEL)                          112
220   CONTINUE
C     LOOP 140 WRITES GROUPS OF NR1 RECORDS EACH ON EACH OF THE WORK
C     TAPES AT THE SAME TIME WRITING EACH RECORD ON THE OUTPUT TAPE.
      IF(NWTP.EQ.0) GO TO 160
      DO 140 I=1,NWTP
      NTP=NTPW(I)
      DO 150 IREC=1,NR1
      READ(NTAPI) IR,(W(IEL,IREC),IEL=1,NEL)                          129
      JREC=JREC+1
150   WRITE(NTAPO) JREC,(W(IEL,IREC),IEL=1,NEL)                       135
      WRITE(NTP)((W(IEL,IREC),IEL=1,NEL),IREC=1,NR1)                  140
140   REWIND NTP
C     LOOP 170 WRITES THE NEXT NR2 RECORDS FROM NTAPI ON NTAPO.
C                                                                     148
160   DO 170 IREC=1,NR2
      READ(NTAPI) IR,(W(IEL,IREC),IEL=1,NEL)                          154
      JREC=JREC+1
170   WRITE(NTAPO) JREC,(W(IEL,IREC),IEL=1,NEL)
C     THE LAST RECORD OF INPUT IS NOW READ AND WRITTEN ON NTAPO.
C                                                                     160
      READ(NTAPI) IR,(W(IEL,NR1),IEL=1,NEL)                           166
      JREC=JREC+1
      WRITE(NTAPO) JREC,(W(IEL,NR1),IEL=1,NEL)
C     NOW NR2 RECORDS STORED IN W ARE WRITTEN ON NTAPO IN REVERSE ORDER.
C                                                                     174
      DO 180 IREC=1,NR2
      IR=NR2-IREC+1
      JREC=JREC+1
180   WRITE(NTAPO) JREC,(W(IEL,IR),IEL=1,NEL)                         185
      IF(NWTP.EQ.0) GO TO 190
C     DO 200 WRITES THE RECORDS FROM WORK TAPES ON NTAPO IN REVERSE ORD.
      DO 200 I=1,NWTP
      J=NWTP-I+1
      NTP=NTPW(J)
      READ(NTP)((W(IEL,IREC),IEL=1,NEL),IREC=1,NR1)                   200
      DO 210 IREC=1,NR1
      IR=NR1-IREC+1
      JREC=JREC+1
210   WRITE(NTAPO)JREC,(W(IEL,IR),IEL=1,NEL)                          211
200   REWIND NTP                                                      217
190   CONTINUE
      PRINT 600,JREC                                                  220
```

```
600    FORMAT(10X,5HJREC=,I5)
       RETURN
20     PRINT 500                                                    221
500    FORMAT(10X,26HARRAY TCC LARGE TO HANDLE.)
       RETURN
       END
```

D. Packing and Centering for Image Reconstruction

1. NAME

Deck name: PICDM

Entries: PICDIM and PICWRM

2. PURPOSE

To pack integer picture data (between 0 and 63) such that a specified number of frames will be centered on a 1024 x 1024 pixel area (for display) in the case of entry PICDIM. In the case of entry PICWRM, the frames will be centered only in the horizontal direction so that the tape prepared can be used for writing on a film with specified width and writing interval.

3. CALLING SEQUENCE

```
CALL PICDIM (IB, IW, IWP)
```

or

```
CALL PICWRM (IB, IW, IWP)
```

where IB, IW and IWP are work arrays to be dimensioned as described in the listing at the end of this section.

4. INPUT-OUTPUT

4.1 Input: Input data should be on logical unit NTPI (see paragraph 7.3 below). Each input record is assumed to consist of the record number followed by NC integer words of data where NC is defined through a common block (paragraph 7.3). The input records should be in FORTRAN binary form.

4.2 Output: The output data will be written on logical unit NTPO in non-FORTRAN binary format. Under entry PICDIM, an end of file mark is written at the end of each file of output. Under entry PICWRM, no EOF is written by the program. Note that the buffer size for unit NTPO should be at least NDM where NDM is defined in Section 9.

4.3 File storage: The number of work tapes needed equals NFH whenever NFH > 1. No work tapes are needed if NFH = 1. When NFH < 1, the first NFH units which are not the same as NTAPI or NTAPO will be chosen out of the logical units 2, 3, 4, 8, 10, 11, 12, 13, 14, 15. The mnemonics (B3, A4, ...) corresponding to the work tapes needed will be printed on the line printer and the program will pause to enable the operator to provide the work tapes.

5. EXITS

If the data requested to be packed exceeds the available screen area (under entry PICDIM) or the film width (under entry PICWRM), then the program exits giving the on-line and off-line message "REQUESTED PICTURE EXCEEDS ALLOTTED FILM OR SCREEN AREA." If the number n of horizontal frames requested to be packed is greater than 8, the program exits with the message "NUMBER OF HORIZONTAL FRAMES SHD BE .LE. 8. BUT IT IS GIVEN TO BE n."

6. USAGE

The program is written in FORTRAN IV and presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC, SKFBIN, WRITER

7.2 Other programs called: RQSTWT, SVSCI, PCKINT, RWNDWT. The listings for RQSTWT and RWNDWT are attached at the end of Section B. The program SVSCI sets all components of an integer vector equal to a given integer. The routine PCKINT packs the last six bits from each of the words of a six-word array to form a single word.

7.3 External storage used: The common block

```
COMMON/IPICD/NFILE,NR,NC,NELVER,NELHOR,NFV,NFH,
             NTPI,NTPO,IEF,ICOMP,FILMW,WRINT,FRAMAR
```

supplies the input parameters needed for the programs.

NFILE = Number of output files needed.

NR = Number of rows (records) in each of the frames to be packed.

NC = Number of columns (unpacked words per record) in each of the frames to be packed.

NELVER = Number of times each of the rows is to be repeated while packing.

NELHOR = Number of times each of the pixels is to be repeated in a packed record.

NFV = Number of vertical frames to be displayed.

NFH = Number of horizontal frames to be displayed.

NTPI = Logical unit number of the input.

NTPO = Logical unit number of the output tape.

IEF = 1 if the input frames are separated by EOF's.

= other integer if the input frames are not separated by EOF's.

ICOMP = 0 if the true (positive) image is to be packed.

= 1 if the complement (negative) image is to be packed.

FILMW = Width of film in millimeters.

WRINT = Writing interval in microns.

FRAMAR = Vertical margins needed between frames in millimeters.

The last three parameters need be used only for PICWRM.

Note: The number of records and the number of words per record in each of the frames to be packed during one call should be the same for all the NFILE*NFH*NFV frames of data.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 1163_8 locations of core.

8.2 Execution time: The execution time depends on the sizes of picture data arrays to be packed. If NFH (number of frames in the horizontal direction) equals 1, the program involves only packing of data and writing it on the output tape. If NFH > 1 then the data will be written on work tapes, the work tapes are rewound and then the output records are written. Thus, for a given product NFH*NFV, NFH = 1 results in faster implementation than NFH > 1. (Of course the desired value of NFH depends on the user's needs.) Two typical times for the case NFH = 1 are shown below:

NFILE = NELVER = NELHOR = NFV = NFH = 1 in both cases.

For NR = 900, NC = 850, PICDIM was executed in approximately 170 seconds.

For NR = 1707, NC = 366, FILMW = 25.0, WRINT = 12.5 and FRAMAR = 0.0, PICWRM was executed in approximately 130 seconds.

8.3 I/O load: None

8.4 Restrictions: NFH $\leq$ 8 (due to limit on number of available work tapes)

9. METHOD

9.1 Entry PICDIM: The number of packed words needed to produce one line of display is 171, corresponding to 1026 unpacked words. Using this fact, the horizontal margin available is computed in terms of number of packed words. The margin is equally distributed between frames and the two ends. Similarly the vertical margin available is computed as (1024 - NR*NELVER*NFV) and distributed equally between frames and the two ends. Denote by MARGVE the vertical margin between the frames. Denote by MARGHO the horizontal margin between the frames.

The first NFV frames of the picture are packed and written on the first work tape, the second NFV frames on the second tape, and so on.

Each record on the Ith work tape will have the Jth word equal to zero for all J such that

$$J \notin [\, I * MARGHO + (I-1) + 1,\ I * MARGHO + NHOR \,]$$

where

$$NHOR = (NC * NELHOR)/6$$

After writing the packed records corresponding to NFV * NFH frames on the NFH work tapes, the work tapes are rewound. The Ith record from each of the work tapes is read and the NFH such records are added to form the Ith record of output, for I = 1, 2, 3, ...

If ICOMP = 1, the data values are subtracted by 63 before packing.

9.2 Entry PICWRM: In this case the procedure is the same as in entry PICDIM except that

```
MARGVE = 1000.*FRAMAR/WRINT
```

and the number of packed words per output record is given by

```
NDM = 1000.*FILMW/(WRITN*6.) + 1
```

and the total horizontal margin is computed using NDM instead of 171.

10. COMMENTS

The margin areas appear light (0 density) regardless of whether ICOMP = 0 or 1.

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

The program has been tested for a large number of cases including test patterns and found to work satisfactorily. The pictures shown in this report were all prepared using this program.

```
      SUBROUTINE PICDIM(IB,IW,IWP)
      COMMON /IPICD/NFILE,NP,NC,NELVER,NELHOR,NFV,NFH,NIPI,NIPO,I,F,ICOM
     1P,FILMW,WRINT,FRAMAR
      DIMENSION NREST(2),NIPW(10)
C
C     TO DISPLAY NFV*NFH FRAMES OF NP*NC PIXELS EACH SIMULTANEOUSLY ON
C     THE SCREEN , EACH PIXEL BEING REPEATED NELVER*NELHOR TIMES.
C     IWP SHOULD BE 1 IF INPUT PICTURES ARE SEPARATED BY EOF-S.
C     IF (ICOMP.EQ.1) THE COMPLEMENT OF THE PICTURE (63-PIC. DATA) WILL
C     BE WRITTEN.
C     FILMW=WIDTH OF FILM AVAILABLE FOR WRITING IN MM. WRINT=WRITING INT
C     ERVAL IN MICRONS. FRAMAR=LONGITUDINAL MARGIN BETWEEN FRAMES IN MM.
C     NIPI=INPUT TAPE UNIT, NIPO=OUTPUT TAPE UNIT.
C     IB SHOULD BE 0-NO. OF LOST BUM WHERE NUM IS 171 IF PICDIM IS CALLED
C     AND NUM=(100*.95*FILMW/(WRINT*5.))+1. IF PICWRM IS CALLED. IW SHO BE
C     0-FOR MAX(NCOM,NC). IWP SHO BE 0-NO NC*NELHOR.
C     AT THE END OF EVERY FILE OF OUTPUT EOF IS WRITTEN WHEN PICDIM IS A
C     CALLED.
C     NO EOF IS WRITTEN ON OUTPUT TAPE WHEN PICWRM IS CALLED.
C
      DIMENSION IB(1),IW(1),IWP(1)
      IPLA=0
      NC=171
      IF(IB.LT.1)G
      CALL PICWRM(IB,IW,IWP)
      NC=(.95*FILMW/(WRINT*5.))+1.
      I=100+1
      FRAMAR=FRAMAR*100./WRINT
  100 CONTINU
      NIPI=NIPI
      NIPO=NIPO
      I=IB+1
C
C     FILM AND PRINT TAPE UNITS READED.
C
      IF(NFH.GT.0)GO TO 2000
      NREST(1)=NIPI
      NREST(2)=NIPO
      IF(NFH.GT.1)CALL RESTWT(NFH,NIPW,2,NREST)
C
C     COMPUTE MARGINS.
C                                                                    15
      IF(IPLAG.NE.1) GO TO 220
      MARGV=(1024-NR*, LVER*NFV)
      IF(MARGV.LT.0) GO TO 1000
      MARGV=MARGV/(NFV+1)
  220 CONTINUE
      NIPW=(NC*NELHOR)/6
      MARGH=NUM-NFHR*NFH
      IF(MARGH.LT.0) GO TO 1000
      MARGH=MARGH/(NFH+1)
      DO 1 IFILE=1,NFILE
C
C     LOOP 20 WRITES THE GROUPS OF PICTURES, EACH GROUP WITH NFV*NFH'S.
C
```

```
      DO 20 IFH=1,NFH
      NTAWO=NTAPW(IFH)
      IF(NFH.GT.1) NTAWO=NTAPO
C
C     FIRST COMPUTE THE BEGINNING AND END THE IB ARRAY CORRESPONDING
C     TO THE IFH-TH COLUMN OF PICTURES.
C
      IC1=(IFH-1)*MARGH+(IFH-1)*NHOR+1
      IC2=IC1+NHOR-1
C
C     LOOP TO WRITE NFV PICTURES ONE BELOW THE OTHER.
C
      DO 30 IFV=1,NFV
      CALL SVBUF(IP,NDM,0)
C
C     WRITE THE VERTICAL MARGIN.
C
      IF(MARGV.EQ.0) GO TO 40                                          44
      DO 50 IR=1,MARGV
      IF(NFH.EQ.1) CALL WRITER(NTAWO,2,IERR,NDM,IB)                    52
      IF(NFH.GT.1) WRITE(NTAWO)(IB(IEL),IEL=1,NDM)                     54
50    CONTINUE
40    CONTINUE
C
C     LOOP TO WRITE ONE PICTURE FRAME.
C
      DO 70 IR=1,NR
      READ (NTAPI) IR=0,(IW(IEL),IEL=1,NC)
C
C     TRANSFER IW TO IWP REPEATING EACH ELEMENT NHOR TIMES.
C                                                                      64
      DO 80 IC=1,NC
      ICC1=MARGH+(IC-1)*(NHOR+1)+1
      ICC2=ICC1+NHOR-1
      IF(ICC1.GT.1) IW(IC)=IW(IC)-IW(IC)
      DO 90 ICC=ICC1,ICC2
      IWP(ICC)=IW(IC)
80    CONTINUE
C
C     PICK IWP AND IB.
C
      JC=1
      DO 100 IC=IC1,IC2
      CALL PCKINT(IB(IC),IWP(JC))                                      90
      JC=JC+6
100   CONTINUE
C
C     WRITE PICKED RECORD, IR, NHOR TIMES.
C
      DO 110 IHOR=1,NHOR
      IF(IFH.EQ.1) CALL WRITER(NTAWO,2,IERR,NDM,IB)                    98
      IF(IFH.GT.1) WRITE(NTAWO) (IB(IEL),IEL=1,NDM)                    100
110   CONTINUE
70    CONTINUE
C
C     THE IFH-TH WORK TAPE HAS BEEN WRITTEN ON THE IFH-TH WORK TAPE.
```

```
C
      IF(IEOF.EQ.1) CALL SKFBIN(NTAPI,1,IERR)                          111
30    CONTINUE
C
C     ONE COLUMN OF NFV FRAMES HAS BEEN PACKED AND WRITTEN ON THE IFH-TH
C     WORK TAPE.
C
      IF(NFH.EQ.1)GO TO 180
      END FILE NTAWO                                                   117
      REWIND NTAWC                                                     118
20    CONTINUE
      IF(NFH.EQ.1) GO TO 10
      NREC=(MARGVE+NR*NELVER)*NFV
      DO 140 IREC=1,NREC
      CALL SVSCI(IB,NCM,0)                                             127
      DO 160 IFH=1,NFH
      NTAWC=NTPW(IFH)
      READ(NTAWC)(IW(IEL),IEL=1,NCM)                                   132
      DO 170 IEL=1,NCM
170   IB(IEL)=IB(IEL)+IW(IEL)
160   CONTINUE
      CALL WRITER(NTAPO,2,ISRR,NCM,IB)                                 146
140   CONTINUE
      CALL RWNDNT(NTPW,NFH)                                            149
180   IF(IFLAG.EQ.0)END FILE NTAPO
C
C     ONE SCREENFUL OF DISPLAY HAS BEEN PREPARED.
C                                                                      151
10    CONTINUE
C
C     NFILE SCREENFULS OF DISPLAY HAVE BEEN PREPARED.
C
      RETURN
1000  PRINT 1100                                                       156
      WRITE(6,1100)                                                    157
1100  FORMAT(/,10X,54HREQUESTED PICTURE EXCEEDS ALLOTTED FILM OR SCREEN
     1AREA)
      RETURN
2000  PRINT 2100,NFH                                                   158
      WRITE(6,2100)NFH                                                 159
2100  FORMAT(/,10X,64HNUMBER OF HORIZONTAL FRAMES SHD BE .LE. 8. BUT IT
     1IS GIVEN TO BE,I3)
      RETURN
      END
```

SECTION III. IMAGE PROCESSING PROGRAMS

A1. Filter Design by Helms' 4-T's Method

1. NAME

Deck name: HELM

Subroutine name: HELMS

2. PURPOSE

This subroutine generates a finite number of filter weights which approximately yield a specified frequency response function. Helms' 4-T's method is used (hence the name).

3. CALLING SEQUENCE

CALL HELMS (IFLAG,N,SAFR,TIMR,L,LMAX,ERROR)

where

IFLAG = An integer between 1 and 6 indicating the type of filter. (1: Low Pass; 2: High Pass; 3: Band Pass; 4: Low Emphasis; 5: High Emphasis; 6: Band Emphasis)

N = Number of samples of the frequency response.

SAFR = Complex output array dimensioned N which will contain the frequency response samples.

TIMR = Complex array dimensioned N which handles the impulse response (i.e., filter weights) corresponding to SAFR.

L = Input constant which gives the initial value of the number of filter weights with which the approximation to desired frequency response is to be attempted. Also, output giving the final number of filter weights used for the approximation.

LMAX = Input constant which gives the upper limit on the number of filter weights.

ERROR = Permissable error = $\max_{i=1,\ldots,N} |G_i - G_i'|$; (an input constant) where G_i and G_i' are the ith components of the given and the approximating frequency responses, respectively.

4. INPUT-OUTPUT

4.1 Input: Input to HELMS is through the calling sequence only.

4.2 Output: L, ISMALL and the first L components of the real part of TIMR are written on logical unit 11.

L = Number of filter weights (integer).

ISMALL = An integer representing the shift in the approximating impulse response (see Section 9).

TIMR = Complex array approximating impulse response.

L, ISMALL are written as one record and Real parts of the components of TIMR are written as L records of one word each.

For each attempted value of L, the value of L and ERR, the error in approximation are printed off-line so that the user knows whether the program stopped with L = LMAX and/or with ERR < ERROR.

4.3 File storage: None

5. EXITS

The program exits when ERR < ERROR or L $\geq$ LMAX during the iterations (see Section 9). If the exit occurs due to the latter condition, L will be set equal to LMAX and the corresponding impulse response will be written on unit 11.

6. USAGE

This program will run on any computer system with FORTRAN IV or a higher level of FORTRAN. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC

7.2 Other programs called: SAMPLE, TRIM (Sections A3 and A4) and FFOURT (a Fast Fourier Transform routine)

7.3 External storage used: Common blocks

```
/WOR/WORK(1024),WORK1(1024)

/PARMTR/OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2,A
```

Common block /WOR/ is for complex work arrays which can be made common with the main program (see Section 10).

OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2,A are constant parameters defining the frequency response to be approximated.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 10505_8 locations of core.

8.2 Execution time: The execution time depends on the number of samples (N) and the number of trials required to obtain the desired approximation to the transfer function. Since the filter lengths are doubled at each successive trial and the length is bounded by LMAX, the number of trials is less than or equal to $\log_2(LMAX/L)$ where L is the initial filter length specified. It has been found that the average time per trial is 169 seconds where N = 512 and it is approximately equal to 43 seconds when N = 256.

8.2 I/O load: The maximum number of lines of (print) output due to HELMS is the smallest integer greater than $\log_2(LMAX/L)$ where L refers to the input value of L.

8.4 Restrictions: $N \leq 1024$ and (Input L) $LMAX \leq 1024$

Dimensions of SAFR and TIMR in the calling program should be greater than or equal to N.

(Even though the program works for all $LMAX \leq 1024$, it is advisable to use $LMAX \leq 310$ so that the filter obtained can be used in conjunction with FFTCNV.)

9. METHOD

Helms' 4-T's method (Transform-Truncate-Transform-Test) is used for approximating the assumed frequency response.

a. The assumed frequency response is sampled and N equally spaced (complex) samples G_n are generated as described under paragraph 9 of the documentation of subroutine SAMPLE (Section A3). For the purposes of filtering, it may be assumed that the weights desired are real numbers and hence the frequency response assumed must have complex conjugate symmetry. That is, $G(-\omega) = G^*(\omega)$ for all ω.

b. The inverse DFT $\{g_n\}$ of the sequence $\{G_n\}$ is found.

c. An integer i is chosen such that

$$0 \leq i \leq N-1$$

and

$$E = \sum_{k=i}^{m} |g_k|$$

is the minimum where

$$m \equiv (i+N-L-1) \pmod{N}$$

d. A new sequence $\{g'_n\}$ is defined as follows

$$g'_n = 0 \quad \text{for} \quad n = i, i+1, \ldots, i+N-L-1 \pmod{N}$$

and

$$g'_n = \mathrm{Re}(g_n) \quad \text{otherwise}$$

The real parts of g_n are taken in the above equation since the filter weights needed are real and the sequence $\{g_n\}$ has, in general, nonzero imaginary parts due to computational errors.

e. The DFT $\{G'_n\}$ of $\{g'_n\}$ is found and the error E_1 in approximating G'_n is computed, where

$$E_1 = \max_{0 \le n \le N-1} \left| G_n - g'_n \right|$$

It is easy to see from the definition of DFT that

$$E_1 = \max_{0 \le n \le N-1} \left| \sum_{k=0}^{N-1} (g_k - G'_k) W^{nk} \right|$$

$$= \max_{0 \le n \le N-1} \left| \sum_{k=i}^{m} g_k W^{nk} + \sum_{k \notin [i,m]} \mathrm{Im}(g_k) W^{nk} \right|$$

$$\max_{0 \le n \le N-1} \left[\sum_{k=i}^{m} |g_k| \, |W^{nk}| + \sum_{k=0}^{N-1} |\mathrm{Im}(g_k)| \, |W^{nk}| \right]$$

where $W = \exp(-2 \, j/N)$

Therefore,

$$E_1 \leq \underset{0 \leq n \leq N-1}{\text{Max}} \left[\sum_{k=i}^{m} |g_k| + \sum_{k=0}^{N-1} |\text{Im}(g_k)| \right]$$

i.e., $$E_1 \leq E + \sum_{k=0}^{N-1} |\text{Im}(g_k)|$$

Hence, the minimization of E defined in step (a) implies minimization of an upper bound on the maximum error in approximating the frequency response.

f. Having computed E_1, it is checked whether E_1 < ERROR (the pre-specified permissible error) or L $\geq$ LMAX. If neither of the conditions holds, then the value of L is doubled and the steps starting from (c) are repeated. If L becomes greater than LMAX when L is doubled, then L is set equal to LMAX and the steps starting from (c) are repeated.

g. Thus, at the conclusion of step f either E_1 < ERROR or L = LMAX (or both). Next, a new sequence $\{g''_n\}$ is defined by shifting all the (N-L) contiguous zero values of the sequence $\{g'_n\}$ to the last (N-L) positions. That is,

$$\left.\begin{aligned} g''_0 &= g'_{i+N-L} \\ g''_1 &= g'_{i+N-L+1} \\ &\vdots \\ g''_{L-2} &= g'_{i-2} \\ g''_{L-1} &= g'_{i-1} \end{aligned}\right\} \text{ L terms}$$

$$\left.\begin{aligned} g''_L &= 0 \\ g''_{L+1} &= 0 \\ &\vdots \\ g''_{N-1} &= 0 \end{aligned}\right\} \text{ N-L terms}$$

where the subscripts are taken Modulo N. Now, $g''_0, \ldots, g''_{L-1}$ are the filter weights of a "Finite Impulse Response (FIR)" filter whose frequency response approximates the given frequency response except for a linear phase difference which is completely specified by specifying the number i denoted by ISMALL in the program. Since i indicates the shift in the filter weights (actually, the weights g'_n are shifted forward by L-i subscripts to give weights g''_n), the number i is called a "shift parameter".

h. The values of L, i and $g''_0, \ldots, g''_{L-1}$ are written on tape as the outputs of the subroutine HELMS.

A flow chart is shown in Figure 3.1 showing the various steps in the subroutine.

10. COMMENTS

At the end of a call of HELMS, the work arrays WORK and WORK1 will contain the approximating frequency response and the exact impulse response (i.e., the IDFT of the Sampled Frequency Response), respectively. By including a common block /WOR/ in the calling program WORK and WORK1 can be retrieved if necessary (e.g., to get plots of original and approximating frequency response).

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

The program has been used for generating several low pass, high pass, band pass and low emphasis, high emphasis and band emphasis filters. See Section III.C of Part I for some examples of the original and approximating filter transfer functions.

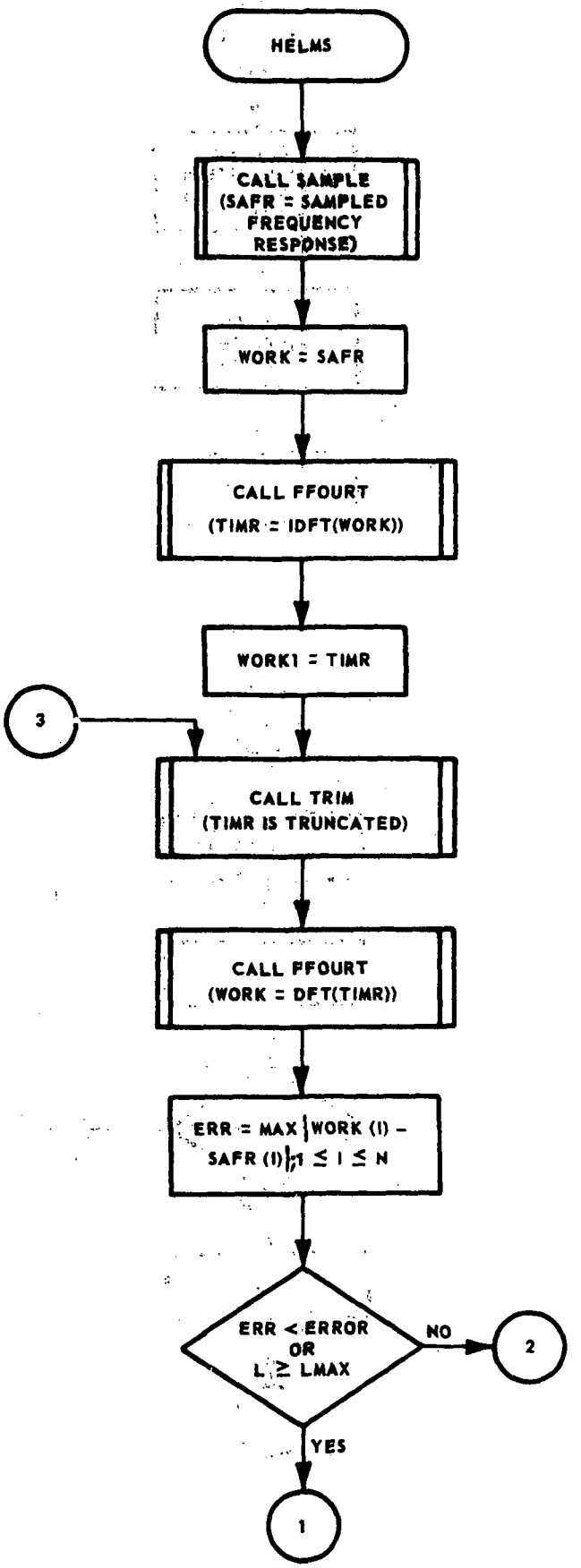


Figure 3.1: Flow Chart for Subroutine HELMS

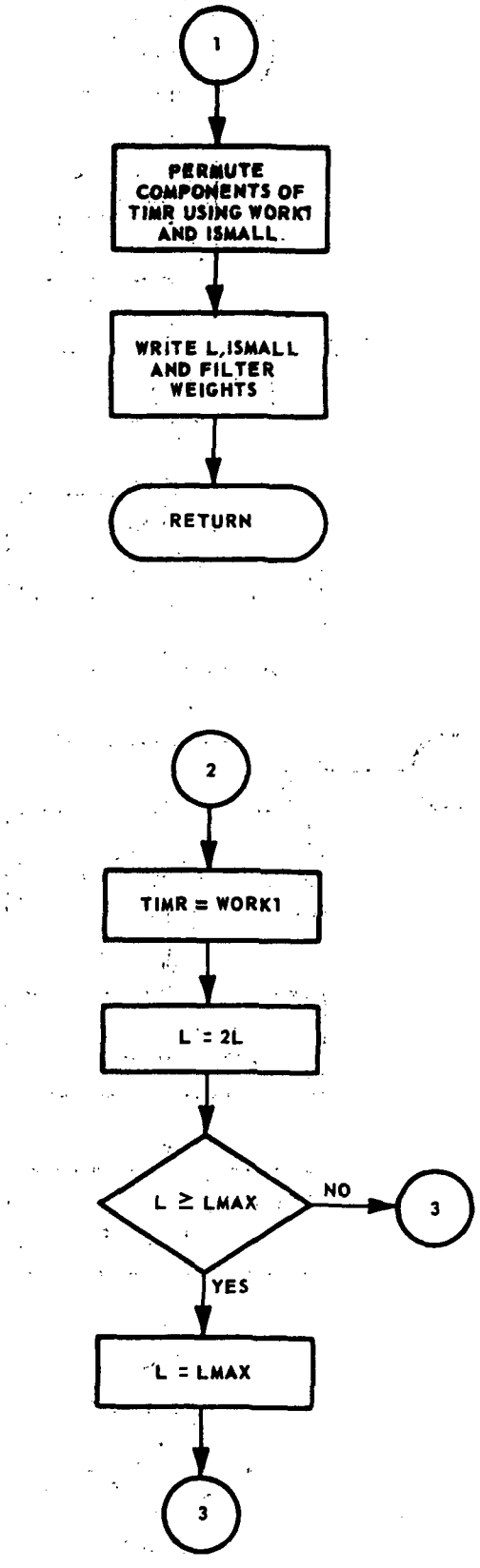


Figure 3.1: Flow Chart for Subroutine HELMS (Continued)

```
      SUBROUTINE HELMS(IFLAG,N,SAFR,TIMR,L,LMAX,ERROR)
C
C     THIS SUBROUTINE GENERATES L IMPULSE RESPONSE WEIGHTS OF A FILTER
C     WHOSE FREQUENCY RESPONSE IS SPECIFIED.  L IS DETERMINED TO GET THE
C     DESIRED APPROXIMATION TO THE FREQUENCY RESPONSE.  LMAX IS SPEC-
C     IFIED SO THAT THE LENGTH OF THE FILTER IS LIMITED BY LMAX EVEN THO
C     THE DESIRED APPROXIMATION CANNOT BE OBTAINED.  IFLAG=1,2 AND 3
C     GIVE LOW,HIGH AND BAND PASS.  IFLAG=4,5,6 GIVE LOW, HIGH AND BAND
C     EMPHASIS.
C
      COMPLEX SAFR(N),TIMR(N)
      COMPLEX WORK,WORK1
C
C     COMMON BLOCK /PARMTR/ HAS PARAMETERS COMMON TO THE MAIN PROGRAM,
C     HELMS AND SAMPLE.
C     COMMON BLOCK /WOR/ HAS WORK ARRAYS WHICH CAN BE OPTIONALLY COMMON
C     WITH THE CALLING MAIN PROGRAM.
C
      COMMON /WOR/WORK(1024),WORK1(1024)
      COMMON /PARMTR/OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2,A
      CALL SAMPLE(IFLAG,N,SAFR)
C
C     SAFR HAS N EQUALLY SPACED SAMPLES OF THE GIVEN FREQUENCY RESPONSE.
C                                                                     2
      DO 301 I=1,N
  301 WORK(I)=SAFR(I)
C
C     SAFR IS STORED IN WORK
C
      CALL FFOURT(WORK,TIMR,N,1.,0)
C
C     TIMR IS THE IMPULSE RESPONSE CORRESPONDING TO SAFR.
C                                                                    12
      DO 50 I=1,N
   50 WORK1(I)=TIMR(I)
C
C     WORK1 HAS THE IDFT OF SAFR.
C
   20 CONTINUE
      CALL TRIM(TIMR,N,L,ISMALL)
C
C     TIMR HAS THE IDFT OF SAFR OPTIMALLY TRUNCATED TO L COMPONENTS.
C                                                                    23
      DO 80 I=1,N
   80 TIMR(I)=CMPLX(REAL(TIMR(I)),0.)
C
C     THE IMAGINARY PARTS OF TIMR ARE SET TO ZERO.
C
      CALL FFOURT(TIMR,WORK,N,-1.,0)
C
C     WORK HAS THE DFT OF TIMR.
C                                                                    33
      ERR=0.
      DO 102 I=1,N
      ERRI=REAL(SAFR(I)-WORK(I))**2+AIMAG(SAFR(I)-WORK(I))**2
```

```
      HELM      - EFN   SOURCE STATEMENT  - IFN(S)  -
      ERRI=ERRI**.5                                                      40
      IF(ERR.LT.ERRI) ERR=ERRI
  102 CONTINUE
C
C     THE ERROR IN APPROXIMATING THE GIVEN FREQUENCY RESPONSE IS FOUND.
C
      WRITE (6,30) L,ERR                                                 45
   30 FORMAT(10X,2HL=,I3,4HERR=,E12.4)
      IF(ERR.LT.ERROR.OR.L.GE.LMAX) GO TO 10
      DO 60 I=1,N
   60 TIMR(I)=WORK1(I)
C
C     IDFT OF SAFR IS RESTORED TO TIMR.
C
      L=2*L
      IF(L.GE.LMAX) L=LMAX
      GO TO 20
   10 CONTINUE
C
C     THE VALUE OF L SATISFYING THE DESIRED ERROR CRITERION OR LIMITED
C     BY LMAX HAS BEEN FOUND.
C
      DO 40 I=1,N
      II=MOD(ISMALL+N-L+I-2,N)+1
      IF(I.LE.L) TIMR(I)=WORK1(II)
      IF(I.GT.L)TIMR(I)=0.
   40 CONTINUE
C
C     THE TRUNCATED IMPULSE RESPONSE IS CYCLICALLY PERMUTED SO THAT
C     THE N-L TERMS SET TO ZERO APPEAR AT THE END.
C
      WRITE(11)L,ISMALL                                                  75
      DO 70 I=1,L
      RTIMR=REAL(TIMR(I))
      TIMR(I)=CMPLX(RTIMR,0.)
   70 WRITE (11)RTIMR
C
C     L, ISMALL AND THE L FILTER WEIGHTS HAVE BEEN WRITTEN ON TAPE 11.
C                                                                        81
      RETURN
      END
```

A2. "Optimum" Filter Design by Helms' 4-T's Method

1. NAME

Deck name: HELM2

Subroutine name: HELMS2

2. PURPOSE

This subroutine generates a filter with the smallest number of filter weights which approximate a specified frequency response number to within a specified error.

3. CALLING SEQUENCE

CALL HELMS2 (IFLAG, N, SAFR, TIMR, L, LMAX, ERROR)

where all the calling arguments except L have the same meaning as in the case of HELMS (Section A1). The argument L is only an output giving the final value of the number of filter weights.

4. INPUT-OUTPUT

Same as in the case of HELMS.

5. EXITS

No abnormal exits.

6. USAGE

The program is written in FORTRAN IV. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

Same as for HELMS.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 1356_8 locations of core

8.2 Execution time: It can be seen from paragraph 9 that the number of trials of filter lengths is less than or equal to $\log_2 (LMAX - 2) + 3$. The time per trial is the same as for HELMS.

8.3 I/O load: Same as for HELMS.

8.4 Restrictions: Same as for HELMS.

9. METHOD

This program differs from HELMS only in the way a new L is defined in step f (paragraph 9, Section A1). Denote by L_i, E_i the values of filter length and error, respectively, during the ith trial. The program tries $L_1 = 1$ and $L_2 = LMAX$. It is assumed that $E_1 > ERROR$. (It is reasonable to assume this because most filters of interest cannot be approximated with just one filter weight. One filter weight is just a constant gain.) If $E_2 > ERROR$, the program writes LMAX filter weights on unit 11 and returns. If $E_2 \leq ERROR$, the following rule is used for defining L_{i+1} from $L_1, L_2, \ldots, L_i$.

$$L_{i+1} = (L_i + L_j)/2$$

where

$$j = \text{Max} \left\{ k < i \;\middle|\; (E_i - ERROR)(E_k - ERROR) \leq 0 \right\}$$

The iterations are stopped when L_{m-1} and L_m have been found such that $L_m - L_{m-1} = 1$. Then, the final value of L will be chosen from the sequence $L_1, \ldots L_m$ such that the smallest value of L_i with $E_i \leq ERROR$ is obtained.

It is quite simple to verify that the number m of iterations needed is limited by

$$m \leq \log_2 \left| LMAX - 2 \right| + 3$$

10. COMMENTS

Same as for HELMS.

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

This program was tested by designing several band pass and band emphasis filters. With N = 256, the number of iterations was found to be equal to 10.

HELM2 - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE HELMS2(IFLAG,N,SAFR,TIMR,L,LMAX,ERROR)
C
C     THIS SUBROUTINE GENERATES L IMPULSE RESPONSE WEIGHTS OF A FILTER
C     WHOSE FREQUENCY RESPONSE IS SPECIFIED.  L IS DETERMINED TO GET THE
C     DESIRED APPROXIMATION TO THE FREQUENCY RESPONSE.  LMAX IS SPEC-
C     IFIED SO THAT THE LENGTH OF THE FILTER IS LIMITED BY LMAX EVEN THO
C     THE DESIRED APPROXIMATION CANNOT BE OBTAINED.  IFLAG=1,2 AND 3
C     GIVE LOW,HIGH AND BAND PASS.  IFLAG=4,5,6 GIVE LOW, HIGH AND BAND
C     EMPHASIS.
C
      COMPLEX SAFR(N),TIMR(N)
      DIMENSION ISMA(60),LL(60),ER(60)
      COMPLEX WORK,WORK1
C
C     COMMON BLOCK /PARMTR/ HAS PARAMETERS COMMON TO THE MAIN PROGRAM,
C     HELMS AND SAMPLE.
C     COMMON BLOCK /WOR/ HAS WORK ARRAYS WHICH CAN BE OPTIONALLY COMMON
C     WITH THE CALLING MAIN PROGRAM.
C
      COMMON /WOR/WORK(1024),WORK1(1024)
      COMMON /PARMTR/OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2,A
      LOGICAL B
      IRND(A)=IFIX(SIGN(ABS(A)+.5,A))
      CALL SAMPLE(IFLAG,N,SAFR)
C
C     SAFR HAS N EQUALLY SPACED SAMPLES OF THE GIVEN FREQUENCY RESPONSE.
C                                                                     4
      DO 301 I=1,N
301   WORK(I)=SAFR(I)
C
C     SAFR IS STORED IN WORK
C
      CALL FFOURT(WORK,TIMR,N,1.,0)
C
C     TIMR IS THE IMPULSE RESPONSE CORRESPONDING TO SAFR.
C                                                                    14
      DO 50 I=1,N
50    WORK1(I)=TIMR(I)
C
C     WORK1 HAS THE IDFT OF SAFR.
C
      L=1
      ITER=1
20    CONTINUE
      CALL TRIM(TIMR,N,L,ISMALL)
C
C     TIMR HAS THE IDFT OF SAFR OPTIMALLY TRUNCATED TO L COMPONENTS.
C                                                                    26
      DO 80 I=1,N
80    TIMR(I)=CMPLX(REAL(TIMR(I)),0.)
C
C     THE IMAGINARY PARTS OF TIMR ARE SET TO ZERO.
C
      CALL FFOURT(TIMR,WORK,N,-1.,0)
C
```

HELM2 - EFN SOURCE STATEMENT - IFN(S) -.

```
C     WORK HAS THE DFT OF TIMR.
C                                                                      36
      ERR=0.
      DO 102 I=1,N
      ERRI=REAL(SAFR(I)-WORK(I))**2+AIMAG(SAFR(I)-WORK(I))**2
      ERRI=ERRI**.5                                                    43
      IF(ERR.LT.ERRI) ERR=ERRI
102   CONTINUE
      ER(ITER)=ERR
      ISMA(ITER)=ISMALL
      LL(ITER)=L
C
C     THE ERROR IN APPROXIMATING THE GIVEN FREQUENCY RESPONSE IS FOUND.
C
      DO 60 I=1,N
60    TIMR(I)=WORK1(I)
C
C     IDFT OF SAFR IS RESTORED TO TIMR.
C
30    FORMAT(10X,5HITER=,I3,3H L=,I3,5H ERR=,E12.4)
      PRINT 30,ITER,L,ERR                                              59
      WRITE(6,30) ITER,L,ERR                                           60
      IF(ERR.EQ.ERROR) GO TO 10
      IF(ITER.NE.1) GO TO 90
      L=LMAX
      ITER=2
      GO TO 20
90    CONTINUE
      IF(ER(2).GT.ERROR) GO TO 100
      ITER1=ITER-1
      IF(IABS(LL(ITER)-LL(ITER1)).LE.1) GO TO 140
      DO 110 I=1,ITER1
      ITER2=ITER-I
      W=(ER(ITER)-ERROR)*(ER(ITER2)-ERROR)
      IF(W.GT.0.) GO TO 110
      L=(LL(ITER)+LL(ITER2))/2
      ITER=ITER+1
      GO TO 20
110   CONTINUE
100   WRITE(6,400)                                                     95
400   FORMAT(10X,23HERR.GT.ERROR FOR L=LMAX)
      GO TO 10
140   CONTINUE
      J1=0
      J2=0
      ERMN=1.E10
      ERMN1=ERMN
      L2=LMAX
      DO 150 I=1,ITER
      IF(ER(I).GE.ERROR.AND.ER(I).LE.ERMN) ERMN1=ER(I)
      IF(ER(I).GE.ERROR.AND.(ER(I).LT.ERMN.OR.ER(I).EQ.ERMN.AND.LL(I).LT
     1.L1))J1=I
      IF(J1.EQ.I) L1=LL(I)
      IF(ER(I).LE.ERROR.AND.LL(I).LT.L2) J2=I
      IF(J2.EQ.I) L2=LL(I)
      ERMN=ERMN1
```

HELM2 - EFN SOURCE STATEMENT - IFN(S) -

```
      ERMX=ERMX1
  150 CONTINUE
      J=J2
      IF(J.EQ.0) J=J1
      L=LL(J)
      ISMALL=ISMA(J)
      ERR=ER(J)
   10 CONTINUE
C
C
      DO 40 I=1,N
      II=MOD(ISMALL+N-L+I-2,N)+1
      IF(I.LE.L) TIMR(I)=WORK1(II)
      IF(I.GT.L)TIMR(I)=0.
   40 CONTINUE
C
C     THE TRUNCATED IMPULSE RESPONSE IS CYCLICALLY PERMUTED SO THAT
C     THE N-L TERMS SET TO ZERO APPEAR AT THE END.
C
      WRITE(11)L,ISMALL                                              150
      DO 70 I=1,L
      RTIMR=REAL(TIMR(I))
      TIMR(I)=CMPLX(RTIMR,0.)
   70 WRITE (11)RTIMR                                                156
      WRITE(6,500) L,ISMALL,ERR                                      158
      PRINT 500,L,ISMALL,ERR                                         159
  500 FORMAT(/,10X,2HL=,I4,8H ISMALL=,I4,5H ERR=,E12.3)
C
C     L, ISMALL AND THE L FILTER WEIGHTS HAVE BEEN WRITTEN ON TAPE 11.
C
      RETURN
      END
```

A3. Sampled Frequency Response Generation

1. NAME

Deck name: SAMPL

Subroutine name: SAMPLE

2. PURPOSE

To generate a given number of equally spaced samples of a specified frequency response function.

3. CALLING SEQUENCE

CALL SAMPLE (IFLAG, N, SAFR)

where

IFLAG = An integer input between 1 and 6 indicating the type of filter. (1: Low Pass; 2: High Pass; 3: Band Pass; 4: Low Emphasis; 5: High Emphasis; 6: Band Emphasis)

N = Number of samples of the frequency response (input integer).

SAFR = Complex output array dimensioned N which will contain the frequency response samples.

4. INPUT-OUTPUT

4.1 Input: Input to SAMPLE is through the calling sequence only.

4.2 Output: Output is through the calling sequence only.

4.3 File storage: None.

5. EXITS

No non-standard exits.

6. USAGE

This program will run on any computer system with FORTRAN IV or a higher level of FORTRAN. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System Subroutines: SYSLOC

7.2 Other programs called: FREDM1, FREDM2, FREDM3, FREDA (See listings at the end of this section)

7.3 External storage used: Common block

```
COMMON/PARMTR/OMEGC1,OMEGC2,OMEGT2,P1,P2,A
```

contains parameters describing the frequency response (to be defined in the calling program).

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 170_8 locations of core

8.2 Execution time: TBD

8.3 I/O load: No input/output commands.

8.4 Restrictions: None

9. METHOD

The expressions for the magnitude of the given band limited frequency response are assumed to be defined by the functions FREDM1, FREDM2, FREDM3 and the phase of the frequency response is assumed to be defined by function FREDA. (FRED ≡ Frequency REsponse Definition; M ≡ Magnitude; A ≡ Angle)

In the present implementation, FREDM1 defines a low pass characteristic $|G_1(\omega)|$ shown in Figure 3.2a. In the interval $[-\omega_s/2,\ \omega_s/2]$,

$$|G_1(\omega)| = 1 \quad \text{for} \quad |\omega| \leq \omega_c$$

$$= 0 \quad \text{for} \quad |\omega| \geq \omega_T$$

and

$$|G_1(\omega)| = \left(\frac{\omega_T - |\omega|}{\omega_T - \omega_c}\right)^p \quad \text{for} \quad \omega_c \leq |\omega| \leq \omega_T$$

where

ω_s = $2\pi f_s$

f_s = sampling frequency = $1/\Delta x$

Δx = sampling interval

ω_c, ω_T and p are specified parameters

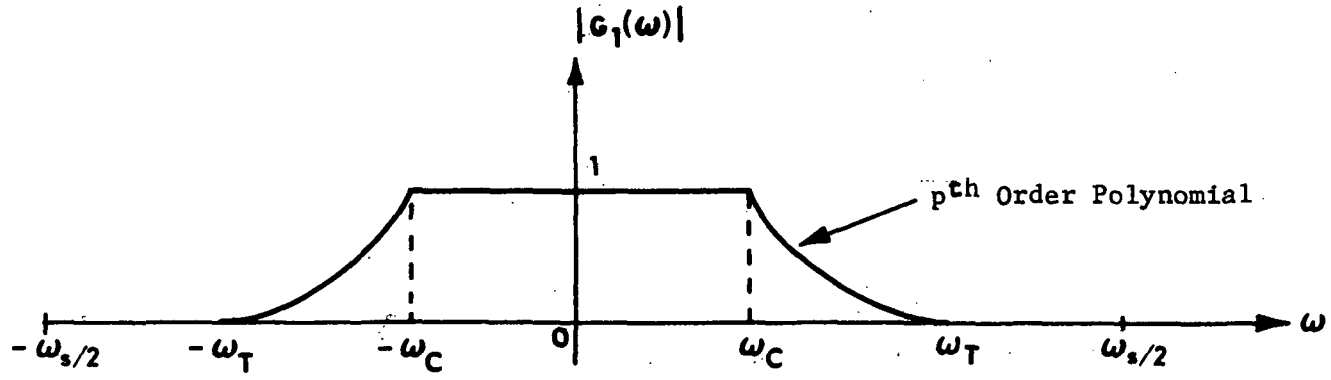


Figure 3.2a Magnitude Characteristic of a Low Pass Filter

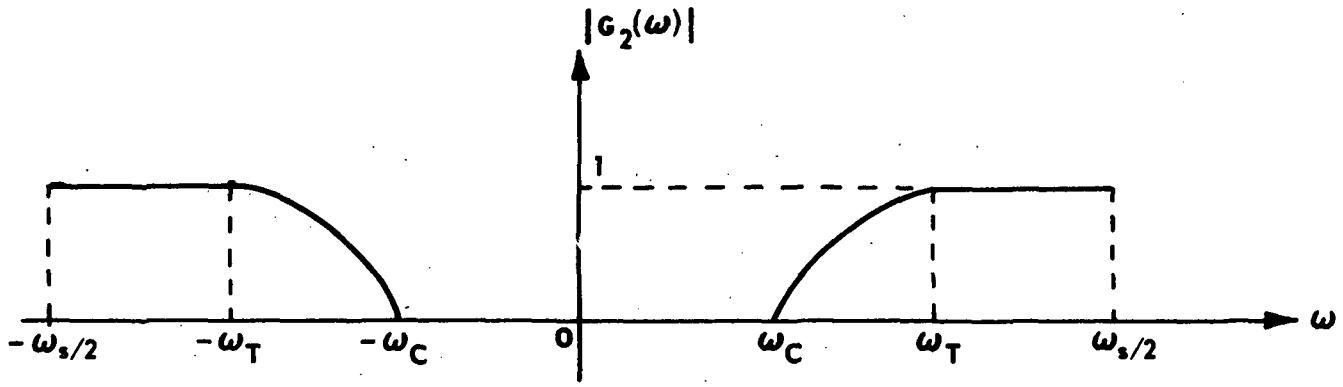


Figure 3.2b Magnitude Characteristic of a High Pass Filter

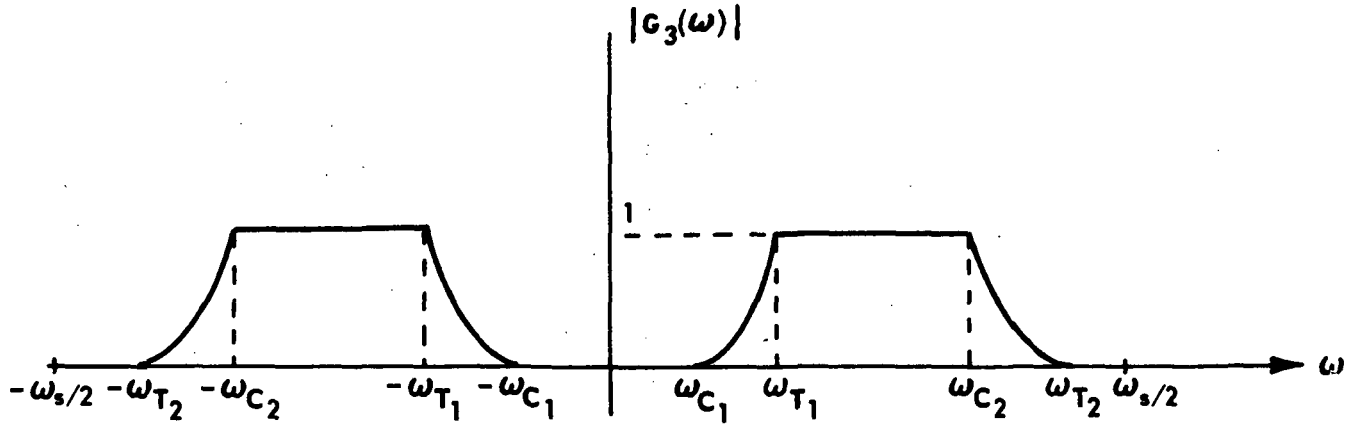


Figure 3.2c Magnitude Characteristic of a Band Pass Filter

Function FREDM2 defines a high pass characteristic $G_2(\omega)$ shown in Figure 3.2b.

$$G_2(\omega) = 1 - G_1(\omega) \quad \text{for}$$

Function FREDM3 defines a band pass characteristic $G_3(\omega)$ shown in Figure 3.2c.

$$G_3(\omega) = G_{12}(\omega) - G_{11}(\omega)$$

where

$$G_{1i}(\omega) = G_1(\omega) \quad \text{with } \omega_c = \omega_{ci},\ \omega_T = \omega_{Ti} \text{ and } p = p_i$$

for i = 1, 2.

It is assumed that the calling program specifies the parameters ω_{c_i}, ω_{T_i} as fractions of ω_s. It is then to be noted that

$$0 \leq \text{OMEGC1} \leq \text{OMEGT1} \leq \text{OMEGC2} \leq \text{OMEGT2} \leq 0.5$$

in the case of the band pass filter. Further, OMEGC1 and OMEGT1 are specified as 0 in the case of low and high pass filters.

Now, sample numbers n = 1, ..., N are assigned to frequencies between $-\omega_s/2$ and $\omega_s/2$ as follows:

$$\omega_n = \left[(n-1)/N\right] \omega_s/2 \qquad \text{for } n = 1, \ldots, N/2$$

and

$$\omega_n = -\left[(N - n+1)/N\right] \omega_s/2 \qquad \text{for } n = N/2 + 1, \ldots, N$$

The SAmpled Frequency Response, SAFR, is defined by

$$\text{SAFR}(n) = \left|G_i(\omega_n)\right| \exp(j\theta(\omega_n))$$

where $\theta(\omega)$ is the phase defined by function FREDA and i is chosen to be 1, 2, 3 for low, high, band pass, respectively.

If IFLAG is specified as 4, 5 or 6, then low, high, or band emphasis filter characteristics are obtained by modifying the corresponding "pass" characteristics. Only the magnitude characteristics are changed such that the "stopband" gain is A instead of zero and the "passband" gain is 1. In fact,

$$SAFR(n) = \left[\left| G_i(\omega_n) \right| (1-A) + A \right] \exp(j\,\theta(\omega_n))$$

where i = IFLAG-3 and the other symbols are as defined above.

10. COMMENTS

The polynomial approximation to the low pass characteristic $G_1(\omega)$ can be replaced by any other desired type of approximation by modifying the function FREDM1 suitably.

11. LISTING

A listing of subroutine SAMPLE and function subprograms FREDM1, FREDM2, FREDM3 and FREDA are shown at the end of this section.

12. TESTS

The program has been tested in conjunction with HELMS. See paragraph 12 under Section A1.

SAMPL - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE SAMPLE(IFLAG,N,SAFR)
C
C     THIS SUBROUTINE PRODUCES A SEQUENCE OF N EQUALLY SPACED SAMPLES
C     SAFR OF THE FREQUENCY RESPONSE.  IFLAG=1,2,3 GIVE LOW,HIGH,BAND
C     PASS RESPECTIVELY.  IFLAG=4,5,6 GIVE LOW,HIGH,BAND+EMPHASIS.
C
      COMPLEX SAFR(N)
C
C     COMMON BLOCK /PARMTR/ HAS PARAMETERS COMMON TO THE MAIN PROGRAM,
C     HELMS AND SAMPLE.
C
      COMMON /PARMTR/OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2,A
      IFLAG1=0
      IF(IFLAG.GT.3)IFLAG1=1
      IFLAG=IFLAG-3*IFLAG1
      DO 30 J=1,N
      IF(J.LE.N/2) OMEG=FLOAT(J-1)/FLOAT(N)
      IF(J.GT.N/2) OMEG=-FLOAT(N-J+1)/FLOAT(N)
      GO TO(11,12,13),IFLAG
11    SAFRR=FREDM1(OMEG,OMEGC2,OMEGT2,P2)*COS(FREDA(OMEG))                15  16  17
      SAFRI=FREDM1(OMEG,OMEGC2,OMEGT2,P2)*SIN(FREDA(OMEG))                18  19  20
      GO TO 10
12    SAFRR=FREDM2(OMEG,OMEGC2,OMEGT2,P2)*COS(FREDA(OMEG))                23  24  25
      SAFRI=FREDM2(OMEG,OMEGC2,OMEGT2,P2)*SIN(FREDA(OMEG))                26  27  28
      GO TO 10
13    SAFRR=FREDM3(OMEG,OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2)*COS(FREDA(
     1OMEG))                                                              31  32  33
      SAFRI=FREDM3(OMEG,OMEGC1,OMEGT1,OMEGC2,OMEGT2,P1,P2)*SIN(FREDA(
     1OMEG))                                                              34  35  36
10    SAFR(J)=CMPLX(SAFRR,SAFRI)
      IF(IFLAG1.EQ.1)SAFR(J)=(SAFR(J)+CMPLX(COS(FREDA(OMEG)),SIN(FREDA(O
     1MEG)))*(A/(1.-A)))*(1.-A)                                           40  41  42
30    CONTINUE
      IFLAG=IFLAG+3*IFLAG1
      RETURN
      END
```

FRDM1 - EFN SOURCE STATEMENT - IFN(S) -

```
      FUNCTION FREDM1(OMEG,OMEGC,OMEGT,P)
C     FREDM1 IS MAGNITUDE RESPONSE OF A LOWPASS FREQ RESPONSE FUNCTION
      IF(ABS(OMEG).GE.ABS(OMEGT)) GO TO 1
      IF(ABS(OMEG).LE.(OMEGC)) GO TO 2
      IF(OMEGC.LT.OMEG.AND.OMEG.LT.OMEGT) GO TO 3
      IF(-OMEGT.LT.OMEG.AND.OMEG.LT.-OMEGC) GO TO 4
1     FREDM1=0.
      RETURN
2     FREDM1=1.
      RETURN
3     FREDM1=((OMEGT-OMEG)/(OMEGT-OMEGC))**P                          17
      RETURN
4     FREDM1=((OMEG+OMEGT)/(OMEGT-OMEGC))**P                          19
      RETURN
      END
```

FRDM2 - EFN SOURCE STATEMENT - IFN(S) -

```
      FUNCTION FREDM2(OMEG,CMEGC,OMEGT,P)
C     FREDM2 IS MAGNITUDE OF A HIGH PASS FREQ RESPONSE FN
      FREDM2=1.-FREDM1(OMEG,OMEGC,OMEGT,P)
      RETURN
      END
```

FRDM3 - EFN SOURCE STATEMENT - IFN(S) -

```
      FUNCTION FREDM3(CMEG,OMEGC1,CMEGT1,CMEGC2,CMEGT2,P1,P2)
C     FREDM3 IS MAGNITUDE OF A BAND PASS FREC RESPONSE FN
C     NOTE THAT CMEGT2.GE.CMEGC2.GE.OMEGT1.GE.OMEGC1
      FREDM3=FREDM1(CMEG,OMEGC2,OMEGT2,P2)-FREDM1(CMEG,OMEGC1,OMEGT1,
     1P1)                                                    2    3
      RETURN
      END
```

FRDA - EFN SOURCE STATEMENT - IFN(S) -

```
      FUNCTION FREDA(OMEG)
C     FREDA IS THE ARGUMENT OF THE FREQ RESPONSE
      FREDA=0.
      RETURN
      END
```

A4. Impulse Response Truncation

1. NAME

Deck name: TRUIM

Subroutine name: TRIM

2. PURPOSE

To truncate the impulse response (or any complex vector with N components) by setting (N - L) contiguous components to zero, considering the impulse response to be periodic with period N. The components to be set equal to zero are chosen so that the sum of their magnitudes is minimized.

3. CALLING SEQUENCE

CALL TRIM (TIMR, N, L, ISMALL)

TIMR = Input vector with N complex components. Also output vector whose (N - L) components are equal to the corresponding components of the input vector.

N = Input integer

L = Input integer equal to the number of components of TIMR to be retained.

ISMALL = Output integer equal to the subscript starting from which (N - L) components of TIMR are set to zero.

4. INPUT-OUTPUT

4.1 Input: Input is through the calling sequence only.

4.2 Output: Output is through the calling sequence only.

4.3 File storage: None.

5. EXITS

No non-standard exits.

6. USAGE

This subroutine will run on any computer system with FORTRAN IV or a higher level of FORTRAN. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC

7.2 Other programs called: None

7.3 External storage used: None

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 335_8 core locations

8.2 Execution time: TBD

8.3 I/O load: No I/O commands

8.4 Restrictions: None

9. METHOD

Let $x_0, \ldots, x_{N-1}$ be the components of the input vector. Then, for k = 0, ..., N-1, the program computes

$$s_k = \sum_{\ell=0}^{L-1} \left| x_{k+\ell} \right|^2$$

where

$$x_{k+\ell} = x_m \quad \text{with } m \equiv (k+\ell) \pmod{N}$$

The value of k for which s_k is minimum is determined. The components $x_i, x_{i+1}, \ldots, x_{i+N-L}$ are set to zero, the subscripts being again taken modulo N.

10. COMMENTS

None.

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

The program has been tested in conjunction with HELMS (see paragraph 12 under Section A1.

TRUIM - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE TRIM(TIMR,N,L,ISMALL)
C
C     THIS SUBROUTINE SETS N-L CONTIGUOUS COMPONENTS OF THE COMPLEX
C     VECTOR TIMR TO ZERO IN AN OPTIMAL MANNER.
C
      LOGICAL A1,A2,A3
      COMPLEX TIMR(N)
C
C
70    SMALL=1000000.
      DO 10 I=1,N
      TIMM=0.
      NL=N-L
C
C     DO LOOP 20 FINDS THE SUM, TIMM, OF THE MAGNITUDES OF (N-L) ELE-
C     MENTS OF THE IMPULSE RESPONSE STARTING WITH THE ITH ELEMENT AND
C     TAKEN CYCLICALLY (MOD N) .
C
      DO 20 J=1,NL
      II=MOD(I+J-1,N)+1
      RTIM=REAL(TIMR(II))**2
      AITIM=AIMAG(TIMR(II))**2
      TIMM=TIMM+(RTIM+AITIM)**.5                                    11
20    CONTINUE
      IF(SMALL.LT.TIMM) GO TO 10
      SMALL=TIMM
      ISMALL=MOD(I,N)+1
C
C     ISMALL IS THE VALUE OF I FOR WHICH TIMM IS THE SMALLEST.
C
10    CONTINUE
C
C
C
      ISMNL=MOD(ISMALL+N-L-2,N)+1
      A1=ISMALL+N-L-1.LE.N
      DO 60 I=1,N
      A2=(ISMALL.LE.I.AND.I.LE.ISMALL+N-L-1
      A3=(1.LE.I.AND.I.LE.ISMNL).OR.(ISMALL.LE.I.AND.I.LE.N)
      A2=(A1.AND.A2).OR.(.NOT.A1.AND.A3)
      IF(A2) TIMR(I)=0.
60    CONTINUE
      RETURN
      END
```

B1. Point Operations - General

1. NAME

SCALE

2. PURPOSE

To scale a set of numbers linearly or nonlinearly so that the data lie between 0 and 63 to make the data suitable for display. The program handles real as well as integer inputs, and as many data files as specified.

3. CALLING SEQUENCE

CALL SCALE (NFILE, NR, NC, IW, RW, PAR, FUNC, IH)

where

NFILE = Number of input files (Integer $\leq$ 20)

NR = Integer input array where NR(I) = number of records in the Ith file.

NC = Integer input array where NC(I) = number of words of picture data per record of Ith file.(It is assumed that each record of Ith file consists of NC(I) + 1 words, the first word being the record number.)

IW and RW are work arrays for handling the input and output records. (See comments in the attached listing.)

PAR is a real array consisting of parameters defining the external scaling function FUNC.

IH is an integer array for storing the histogram of the output data.

4. INPUT-OUTPUT

See paragraph 7 for the definition of variable names used below.

4.1 Input: Input data are on unit NTAPI. The first word in each record is an integer (the record number) and the remaining NEL words are integer or real data. (If real, ITYPE should be specified as 1.) It is assumed by this subroutine that the input data are in FORTRAN binary format.

4.2 Output: The output data appear on unit NTAPO as integers. The record number is written in the beginning of every record and NEL

data words follow. The format is FORTRAN binary. If input files are separated by EOF (i.e., IEOF is given as 1), the output files also will be separated by EOF.

4.3 File storage: None

5. EXITS

No non-standard exits.

6. USAGE

This program will run on any computer system with FORTRAN IV or a higher level of FORTRAN. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC, SKFBIN

7.2 Other programs called: FUNC (an EXTERNAL function to be specified by the user in the form FUNC (PAR, X). See Section 3, above.) and IRHSTG (see Section B2, below).

7.3 External storage used: Three common blocks are used:

```
COMMON/INSCAL/NTAPI,NTAPO,ITYPE,IEOF,IMN,IFLG,IHST
COMMON/FILE/IFILE
COMMON/WSCAL/RWB,RWS,RWM
```

The block /INSCAL/ supplies the input parameters to the program. The meanings of all the variable names are given in the listing attached. The block /FILE/ is useful if the scaling function FUNC is to be made independent on IFILE, which is a do loop index giving the file number being scaled. In that case, the block /FILE/ should be made common to SCALE and FUNC. The block /WSCAL/ consists of work arrays RWB, RWS, RWM which store the maxima, minima and means of the files being scaled (under the option IFLG $\neq$ 1 and IMN $\neq$ 0). This block should be made common to SCALE and the calling program if the maxima, minima and/or mean are to be used again. A typical example of the use of this common block is when it is necessary to scale the same file with several scaling functions, all of which depend upon the maximum and minimum of the input data. In this case, one would call SCALE first with IFLG $\neq$ 0 and IFLG $\neq$ 1 (say IFLG = 2) and perform scaling with the first function and for all the successive functions one would call SCALE with IFLG = 1.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 665_8 locations of core.

8.2 Execution time: The execution time is highly dependent on NFILE, NR, NC, the option used (see comments in listing), and the scaling function

FUNC. For a case where NFILE = 1, the array size was 1707 x 366, linear scaling was used and the option used was such that the minimum and the maximum of input data were found before scaling, it was found that the time taken was 400 seconds. For the same data when linear scaling was used with threshold and the minimum and maximum were assumed to be given, the time taken was 257 seconds.

8.3 I/O load: No printouts on-line or off-line.

8.4 Restrictions: NFILE $\leq$ 20.

9. METHOD

This program has several options as detailed in the comments in the listing attached. The method given below applies to the case when IFLG $\neq$ (0 or 1), IMN $\neq$ 0, and IHST $\neq$ 0.

(i) First, all the data records are read and the maximum, minimum, and mean values of data in each data file are determined separately and stored in work arrays RWB, RWS, and RWM, respectively. (RWB, RWS, and RWM are dimensioned 20 which permits handling of 20 files.)

(ii) The input tape is rewound.

(iii) Each of the files is scaled separately as follows:

When the Ith file is being scaled

```
PAR(1) = RWB(I)

PAR(2) = RWS(I)

PAR(3) = RWM(I)
```

and the output Y corresponding to input X is given by

```
Y = FUNC (PAR, X)
```

The output Y is computed for each data element and rounded off to the nearest integer. If the integer is greater than 63, it is set to 63. If the integer is less than zero, it is set to zero. The number of occurrences of the integer I in a given output record is added to IH(I) as soon as the record has been computed.

10. COMMENTS

The function FUNC is an arbitrary point operation. It can be chosen depending on the particular needs of the user. This approach has been

taken because "the true representation" of picture data is yet an undefined phenomenon. If one has to represent data points which are known to be all of the same sign, it would be advantageous to use a scaling function which is different from the one in the case where the data can take on both positive and negative values. Also, if the data have a very wide range and parts of them are to be displayed with increased contrast and other parts with decreased contrast, a suitable nonlinear function may be defined.

11. LISTING

A listing of the program is attached at the end of this section.

12. TESTS

The program has been tested for a large number of cases and the scaled pictures have been displayed. The filtered pictures shown in this report were all first scaled using this program and then packed using PICDIM or PICWRM (Section II-D).

SCAL - EFN SOURCE STATEMENT - IFN(S) -

```
      SUBROUTINE SCALE(NFILE,NR,NC,IW,RW,PAR,FUNC,IH)
C     NFILE=NUMBER OF FILES TO BE SCALED WITH NR(I)*NC(I) PIXELS IN THE
C     ITH FILE. IW AND RW MAY BE EQUANCED AND SHD BE D*NED MAX(NC(I)),I=1
C    1,...,NFILE. PAR IS AN ARRAY TO STORE PARAMETERS FOR FUNC, AN EXTER
C     NAL FUNCTION GIVING THE SCALING RULE. PAR SHD BE D*NED AT LEAST 3.
C     NTAPI, NTAPO ARE THE INPUT AND OUTPUT UNITS. ITYPE=1 FOR REAL INPU
C     T. ITYPE.NE.1 FOR INTEGER INPUT. IEOF=1 IF INPUT FILES ARE SEPARAT
C     ED BY EOF. OUTPUT FILES ALSO WILL BE SEPARATED BY EOF IN THIS CASE
C     . IF IMN=0 ONLY MAX AND MIN OF DATA WILL BE COMPUTED. IF IMN.NE.0
C     THE MEAN ALSO WILL BE COMPUTED.
C     IF IFLG.EQ.0 MAX,MIN (AND OPTIONALLY THE MEAN) WILL BE COMPUTED
C     AND NO SCALING WILL BE DONE. IF IFLG.EQ.1 ONLY SCALING WILL BE
C     DONE. IF IFLG.NE.0 OR 1 MAX, MIN (AND OPTIONALLY, MEAN) WILL BE FO
C     FOUND AND SCALING WILL BE DONE.
C     IF  IHST.NE.0 THE HISTOGRAM OF RESCALED DATA OF ALL THE NFILE
C     FILES PUT TOGETHER WILL BE PRODUCED IN IH.

      DIMENSION RWM(20)
      DIMENSION RWB(20),RWS(20),PAR(1),IW(1),RW(1)
      DIMENSION IH(64)
      DIMENSION NR(NFILE),NC(NFILE)
      COMMON /INSCAL/NTAPI,NTAPO,ITYPE,IEOF,IMN,IFLG,IHST
      COMMON /FILE/ IFILE
      COMMON /WSCAL/RWB,RWS,RWM
      IRND(A)=IFIX(SIGN(ABS(A)+.5,A))
      IF(IFLG.EQ.1) GO TO 150
      DO 50 IFILE=1,NFILE
      NREC=NR(IFILE)
      NEL=NC(IFILE)
      RWBIG=-1000000.
      RWSMAL=1000000.
      RWMEAN=0.
      DO 10 I=1,NREC
      IF(ITYPE.EQ.1) GO TO 70
      READ (NTAPI) NLINE,(IW(J),J=1,NEL)                              18
      DO 80 J=1,NEL
      RW(J)=IW(J)
80    CONTINUE
      GO TO 100
70    READ(NTAPI) NLINE, (RW(J),J=1,NEL)                              31
100   CONTINUE
      DO 90 J=1,NEL
      IF(RWBIG.LT.RW(J)) RWBIG=RW(J)
      IF(RWSMAL.GT.RW(J)) RWSMAL=RW(J)
90    CONTINUE
      IF(IMN.EQ.0) GO TO 10
      DO 140 J=1,NEL
140   RWMEAN=RWMEAN+RW(J)
10    CONTINUE
      RWB(IFILE)=RWBIG
      RWS(IFILE)=RWSMAL
      RWM(IFILE)=RWMEAN
      IF(IEOF.EQ.1) CALL SKFBIN(NTAPI,I,IERR)                         64
50    CONTINUE
```

```
      IF(IFLG.EQ.0)RETURN
      REWIND NTAPI                                                    70
150   CONTINUE
      DO 160 I=1,64
160   IH(I)=0
      DO 60 IFILE=1,NFILE
      NREC=NR(IFILE)
      NEL=NC(IFILE)
      PAR(1)=RWB(IFILE)
      PAR(2)=RWS(IFILE)
      IF(IMN.NE.0) PAR(3)=RWM(IFILE)
      DO 20 I=1,NREC
      IF(ITYPE.EQ.1) GO TO 110
      READ(NTAPI) NLINE,(IW(J),J=1,NEL)                               95
      DO 120 J=1,NEL
      RW(J)=IW(J)
120   CONTINUE
      GO TO 130
110   READ(NTAPI) NLINE, (RW(J),J=1,NEL)                              108
130   CONTINUE
      DO 30 J=1,NEL
      RWJ=RW(J)
      W=FUNC(PAR,RWJ)                                                 118
      IW(J)=IRND(W)
      IF(IW(J).LT.0) IW(J)=0
      IF(IW(J).GT.63) IW(J)=63
30    CONTINUE
      IF(IHST.NE.0)CALL IRHSTG(IW,NEL,IH)                             131
40    WRITE(NTAPO) NLINE,(IW(J),J=1,NEL)                              133
20    CONTINUE
      IF(IEOF.EQ.1) CALL SKFBIN(NTAPI,1,IERR)                         141
      IF(IEOF.EQ.1) END FILE NTAPO                                    143
60    CONTINUE
      RETURN
      END
```

B2. Point Operations - Non-Negative Integer Input

A set of simple and fast routines using a table look-up method has been developed for performing point operations on non-negative integer input data. The listings of these routines are attached at the end of this section. Their names and purposes are tabulated below.

Name		Purpose	Other Routines Called
Deck	Subroutine		
TBLKP1	TBLKPV	Given a vector IX and a "table array" ITBL, to generate a vector IY such that IY(I) = ITBL(IX(I)+1).	None
TBLKP2	TBLKPM	Given an input array consisting of non-negative integers on tape unit NTAPI, and a "table array" ITBL, to generate an output array on tape unit NTAP), each of whose records is obtained by using TBLKPV on the corresponding input record.	TBLKPV
IRHIST	IRHSTG	Given integer arrays IX and IH to add the histogram of array IX to the array IH.	None
IPHIST	IPHSTG	To find the histogram of a given integer data array consisting of several records.	IRHIST
DISLN	DISLIN	Given the histogram IH, to set up a "table array" ITBL which can be used in routines TBLKPV and TBLKPM using a particular type of density transformation which linearizes the distribution function.	None

The routine DISLIN is a particular transformation which serves to stretch contrast in some pictures in which the density range is not fully

utilized. The above programs can be used with any other density transformation which can be supplied in the form of a "table array" ITBL.

```
      SUBROUTINE TBLKPV(IX,IY,N,ITBL)
C     THISROUTONE GIVES IY(I)=ITBL(IX(I)+1) FOR I=1,...,N
C     ITBL SHD BE D-NED MAX(IX(I)+1) I=1,...,N. IX AND IY MAY BE THE SAM
      DIMENSION IX(N),IY(N),ITBL(1)
      DO 10 I=1,N
      J=IX(I)+1
10    IY(I)=ITBL(J)
      RETURN
      END
```

```
      SUBROUTINE TBLKPM(NREC,NEL,IX,ITBL,NTAPI,NTAPO)
C     THIS ROUTINE READS DATA FROM NTAPI INTO IX, MODIFIES IT USING
C     ITBL AND WRITES IT ON NTAPO. ITBL SHD BE D-NED MAX(IX(I)+1), I=1,
C     ...,NEL FOR ALL RECORDS 1,...,NREC.
      DIMENSION IX(NEL),ITBL(1)
      DO 10 I=1,NREC
      READ(NTAPI)IREC,(IX(IEL),IEL=1,NEL)                          4
      CALL TBLKPV(IX,IX,NEL,ITBL)                                  10
   10 WRITE(NTAPO)IREC,(IX(IEL),IEL=1,NEL)                         11
      RETURN
      END
```

```
      SUBROUTINE IRHSTG(IX,N,IH)
C     THIS SUBOOURINT ADDS TO IH(J) THE NUMBER OF OCCURENCES OF J-1 IN
C     THE VECTOR IX. IH SHD BE D-NED MAX OF IX(I)+1, I=1,...,N.
      DIMENSION IX(N),IH(1)
      DO 10 I=1,N
      J=IX(I)+1
10    IH(J)=IH(J)+1
      RETURN
      END
```

```
      SUBROUTINE IPHSTG(NREC,NEL,IX,IH,N,NTAPI)
C     TO GET HISTOGRAM OF AN INTEGER FILE CONTAINING NREC RECORDS WITH
C     NEL WORDS EACH. DATA RANGE-- 0 TO M WHERE M.LE.N-1.
      DIMENSION IX(NEL),IH(N)
      DO 10 I=1,N
10    IH(I)=0
      DO 20 I=1,NREC
      READ(NTAPI)IREC,(IX(IEL),IEL=1,NEL)                              11
20    CALL IRHSTG(IX,NEL,IH)                                           18
      RETURN
      END
```

```
      SUBROUTINE DISLIN(IH,ITBL,N)
C     THIS ROUTINE SETS UP DISTRIBUTION LINEARIZATION TABLE ITBL GIVEN
C     HISTOGRAM IH WITH N VALUES. IH AND ITBL MAY BE THE SAME.
      DIMENSION IH(N),ITBL(N)
C     CALCULATE DISTRIBUTION IN ITBL.
      ITBL(1)=IH(1)
      DO 10 I=2,N
10    ITBL(I)=IH(I)+ITBL(I-1)
      RMAXD=ITBL(N)
      M=N-1
C     LINEARIZE DISTRIBUTION AND PUT TABLE IN ITBL.
      DO 20 I=1,M
20    ITBL(I)=FLOAT(N*ITBL(I))/RMAXD+.5
      ITBL(N)=N
      RETURN
      END
```

C. Non-Recursive Filters - Frequency Domain Implementation Using Fast Convolution

1. NAME

Deck name: FILT

Subroutine name: FILTER

2. PURPOSE

To filter a given number of data records, each with a given number of words (e.g., picture data), using a filter whose weights are given.

3. CALLING SEQUENCE

CALL FILTER (NREC, NEL, IFLAG, IX, IY)

where

NREC = Number of records (input).

NEL = Number of words in each record (input).

IFLAG = Input integer equal to 1, 2 or 3 depending on how the ends of a record are to be "padded" to avoid "edge effects."

IX = Integer array into which the subroutine reads the data to be filtered. At the end of call, IX will have the last record of input data.

IY = REAL array which handles the filtered data. At the end of call, IY will have the last record of filtered data.

4. INPUT-OUTPUT

4.1 Input:

(i) Unit 11: The number of weights (L) and the shift parameter (ISMALL) are read as one binary record. The filter weights are L real numbers which are read as L binary records.

(ii) Unit 8: IREC, (IX(I), I=1, NEL) are read as one binary record, where IREC is the record number and IX is the IRECth integer data array. The reading takes place NREC times corresponding to IREC = 1, ..., NREC.

4.2 Output:

Unit 13: IREC, (IY(I), I=1, NEL) are written as one binary record where IREC is the record number and IY is the IRECth

filtered data array. The writing takes place NREC times corresponding to IREC = 1, ..., NREC.

4.3 File storage: None.

5. EXITS

No non-standard exits.

6. USAGE

This program will run on any computer system with FORTRAN IV or a higher level of FORTRAN. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC

7.2 Other programs called: FFTCNV (a fast convolution routine which uses Helms' "select-save" method discussed in Section III-B1 of Part I).

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 5233_8 core locations

8.2 Execution time: The execution time depends on the filter size and the data array size. Typical times are listed in Table 3.5 of Part I.

8.3 I/O load: See INPUT-OUTPUT (Section 4).

8.4 Restrictions: The number of filter weights must be less than or equal to 310.

Number of elements (NEL) per record is restricted to ≤ 2644, limited by the core capacity. NEL ≤ 2141 will result in more efficient input/output buffer allocation.

9. METHOD

The routine uses a more basic subroutine called FFTCNV which implements Helms' select-saving method of evaluating convolutions. See Section III-B1 of Part I for the details of array handling.

If $\{x_k\}$, k = 0, ..., N-1 defines an input record, then the purpose of the parameter IFLAG in this program can be explained as follows:

If IFLAG = 1, the sequence $\{x_k\}$ is considered to be periodic with period N. That is, $x_{-1} = x_{N-1}$, $x_N = x_0$, and so on.

If IFLAG = 2, the sequence $\{x_k\}$ is assumed to be symmetric about the end points. That is, $x_{-1} = x_1$, $x_N = x_{N-2}$, and so on.

If IFLAG = 3, the data is assumed to be zero outside the given record.

10. COMMENTS

None.

11. LISTING

A listing of the subroutine is shown at the end of this section.

12. TESTS

See Section III-B1 for examples of filtered test patterns.

```
      SUBROUTINE FILTER(NREC,NEL,IFLAG,IX,IY)
C
C     FILTERS NREC DATA RECORDS, NEL ELEMENTS EACH, ON TAPE 8 USING WEIG
C     ON TAPE 11 AND GIVES THE FILTERED DATA ON TAPE 13
C
      DIMENSION W(310),WX(1024),WY(1024),IX(NEL),IY(NEL)
      REAL IY
      LOGICAL A1,A2,A3,A4,A5
      IRND(A)=IFIX(SIGN(ABS(A)+.5,A))
C
C     READ L, THE NUMBER OF FILTER WTS, ISMALL, THE 'SHIFT PARAMETER',
C     AND W(I), THE FILTER WTS.
C
      READ(11) L,ISMALL                                              3
      DO 70 I=1,L
      READ(11) RTIMR                                                 8
70    W(I)=RTIMR
C
C     FIND NSTEP, THE NUMBER OF STEPS NEEDED TO FILTER ONE RECORD OF
C     DATA (NSTEP=1 IF NEL+L-1.LE.1024)
C
      NSTEP=(NEL-1)/(1024-L+1)+1
      DO 30 J=1,NREC
      READ(8) IREC,(IX(I),I=1,NEL)
C
C     DURING THE ISTEPTH STEP** WX(ISMALL-1)=IX(N2), IY(N2) THRU IY(N3)
C     ARE THE OUTPUTS OBTAINED, NO IS THE NUMBER OF WX ELEMENTS DEFINED,
C     NCL IS THE NUMBER OF IY ELEMENTS OBTAINED AS OUTPUT.
C                                                                    16
      DO 10 ISTEP=1,NSTEP
      N2=(1024-L+1)*(ISTEP-1)+1
      NO=1024
      IF(ISTEP.EQ.NSTEP) NO=NEL-N2+L
      NCL=NO-(L-1)
      N3=N2+NCL-1
C
C     DO LOOP 50 FILLS WX ARRAY WITH APPROPRIATE ELEMENTS FROM THE DATA
C     ARRAY.  IF IFLAG=1, THE DATA IX IS ASSUMED TO BE PERIODIC WITH
C     PERIOD NEL.  IF IFLAG=2, THE DATA IS ASSUMED TO BE SYMMETRIC ABOUT
C     THE END POINTS.  IF IFLAG=3, THE DATA IS ASSUMED TO BE ZERO OUT-
C     SIDE THE GIVEN RECORD.
C
      DO 50 I=1,NO
      A1=IFLAG.EQ.3
      A2=ISTEP.EQ.1
      A3=I.LT.ISMALL-1
      A4=ISTEP.EQ.NSTEP
      A5=I.GT.ISMALL-N2+NEL-1
      II=N2-ISMALL+I+1
      GO TO(51,52,52),IFLAG
C
51    CONTINUE
      II=II+NEL
      II=MOD(II,NEL)
      IF(II.EQ.0) II=NEL
```

```
      GO TO 53
C
52    CONTINUE
      IF(A2.AND.A3) II=ISMALL-I
      IF(A4.AND.A5) II=II-2*(II-NEL)
C
53    CONTINUE
      WX(I)=IX(II)
      IF(A1.AND.(A2.AND.A3.OR.A4.AND.A5))WX(I)=0.
50    CONTINUE
C
C
      CALL FFTCNV(WX,NC,W,L,WY)
C                                                                    63
      DO 60 I=N2,N3
      II=I-N2+1
      IY(II)=WY(II)
60    CONTINUE
      DO 40 I=1,NEL
40    CONTINUE
C
10    CONTINUE
C
      WRITE(13) IREC,(IY(I),I=1,NEL)
C
C                                                                    77
30    CONTINUE
C
      RETURN
      END
```

D. Recursive Filter Implementation

1. NAME

Deck name: TDFL1

Subroutine name: TDFIL1

2. PURPOSE

To implement a "product-type" separable recursive filter which is a low pass filter in the vertical direction and any filter in the horizontal direction.

3. CALLING SEQUENCE

CALL TDFIL1 (NEL, HORFIL, PAR, IX, X, W, Y, W1)

where NEL is the number of words of picture data per input (and output) record and all other arguments are described in the comments in the attached listing.

5. INPUT-OUTPUT

4.1 Input: The data to be filtered should be on logical unit 8 and the data including its reflections in the top and bottom of the picture generated by subroutine REFLC (see Section II-C) should be on logical unit 10. The program assumes that the number of words in each record on units 8 and 10 is NEL + 1, the first word being the record number and the remaining NEL being the data words. The input data may be either real or integer and should be in FORTRAN binary format.

4.2 Output: The output data will be real. It will be written on unit 11 in the same format as the input.

4.3 File storage: None

5. EXITS

No non-standard exits.

6. USAGE

This program has been written in FORTRAN IV. It is presently implemented on IBM 7094.

7. EXTERNAL INTERFACES

7.1 System subroutines: SYSLOC, BSFILE, SQRT

7.2 Other programs called: VADDI, VSADDI, VADDR, VSADDR, VSUBI, VSSUBI, VSUBR, VSSUBR (all entries under deck name VADSUB; see attached listing) RLPFL1 (a recursive low pass filter routine in one dimension; see attached listing).

7.3 External storage used: Two common blocks are used:

```
COMMON/INPAR/NREC,NRECR,ITYPE

COMMON/VERPAR/LR,NORM
```

See comments under listing attached for definitions of variables in the common block.

8. PERFORMANCE SPECIFICATIONS

8.1 Storage: 1430_8 locations of core

8.2 Execution time: The execution time depends on the size of the array to be filtered, the size of the filter (parameters PAR(1) and LR), the horizontal filter routine HORFIL and whether normalization is requested or not. For example, a 1000 x 320 picture array was filtered using a recursive band pass filter routine for HORFIL and LR = 20, PAR(1) = 25 and NORM = .FALSE. in approximately 8.5 minutes. With NORM = .TRUE., the same filtering operation took about 14.2 minutes.

8.3 I/O load: None

8.4 Restrictions: None

9. METHOD

The implementation when NORM = .FALSE. follows that given in Table 3.4 of Part I for "Product" type separable recursive filters which are low pass in the vertical direction. Note that L_1 and L_2 in Table 3.4 correspond to PAR(1) - 1. and LR - 1, respectively, in the program. Also, the division by the constants $2L_1 + 1$ and $2L_2 + 1$ in the horizontal and vertical low pass filter realizations shown in Figures 3.22 and 3.26 of Part I is not performed since most filtered outputs are linearly rescaled before displaying. When NORM = .TRUE., the output is normalized so that only the variations of the input with respect to the neighbors in a (2LR-1)* (2PAR(1)-1) is significant. The implementation of normalization in this case follows that shown in Figure 3.28 (Part I). To get the matched filter output given by Equation (3.53) of Part I, one needs to divide the output of this program by a constant. The value of this constant depends on the subroutine HORFIL. If HORFIL is either RHPFL1 or RBPFL1 (see attached listings), then this constant is

$$(2LR-1)\sqrt{\sum_{k,\ell} g_{k\ell}^2}$$

where g_k are the corresponding filter weights.

10. COMMENTS

None.

11. LISTING

A listing of this subroutine and the subroutines referred to in this section is attached at the end of this section.

12. TESTS

The Figures 3.29, 3.30, 3.31 of Part I show the results of using the routine TDFIL1 with an external routine RBPFL1 in place of HORFIL. This is equivalent to a matched filter which enhances vertical lines. A typical record was taken in all these cases and normalized by dividing it by $(2LR-1)\sqrt{\sum_{k,\ell} g_{k\ell}^2}$. In the noise-free case, it was found that the outputs of the program were exactly equal to the theoretically calculated values. When the template (filter) matched the line in the test pattern with the correct width (13 pixels) the output was unity. Figures 3.3a and b show a record of the noise-free test pattern and a record of the filtered noise-free test pattern. Figure 3.3c is constructed as follows. The locations of the center lines of the wide vertical lines in the test pattern are determined. The filtered output at these points (i.e., when the center line of the template coincides with the center line of the wide vertical lines) is plotted against the line width. Figures 3.4a, b and c show the corresponding plots for the case of a noisy test pattern where the noise is approximately Gaussian with zero mean and a variance of 8. This gives a signal-to-noise ratio of .687 for the line width 13. The corresponding theoretically expected value of output is $1/\sqrt{1 + 1/.687} = .636$. It is seen that this is very close to the value corresponding to line width 13 in Figure 3.4c.

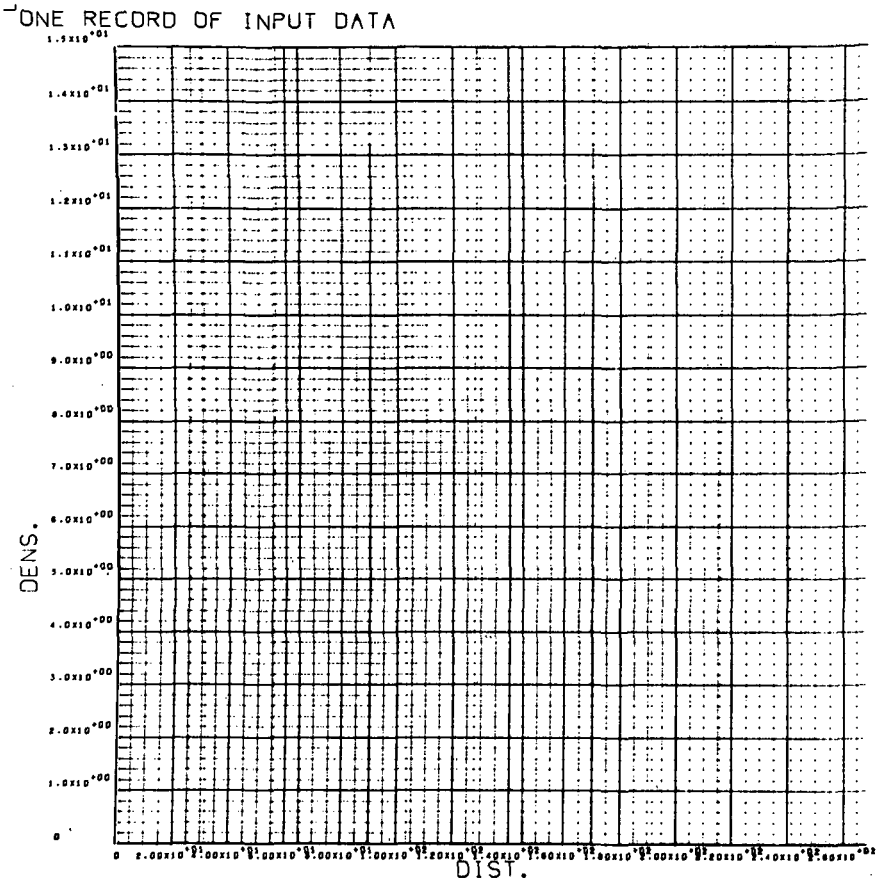


Figure 3.3a One Record of Noise-Free Test Pattern

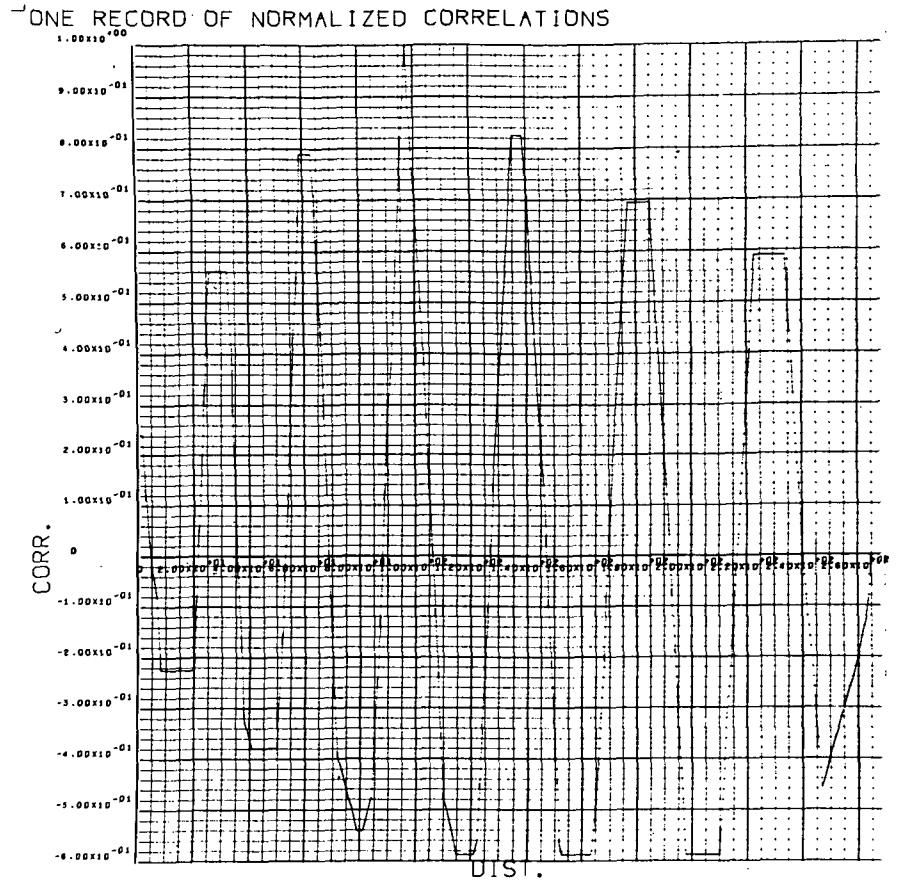


Figure 3.3b One Record of Matched Filtered Noise-Free Test Pattern

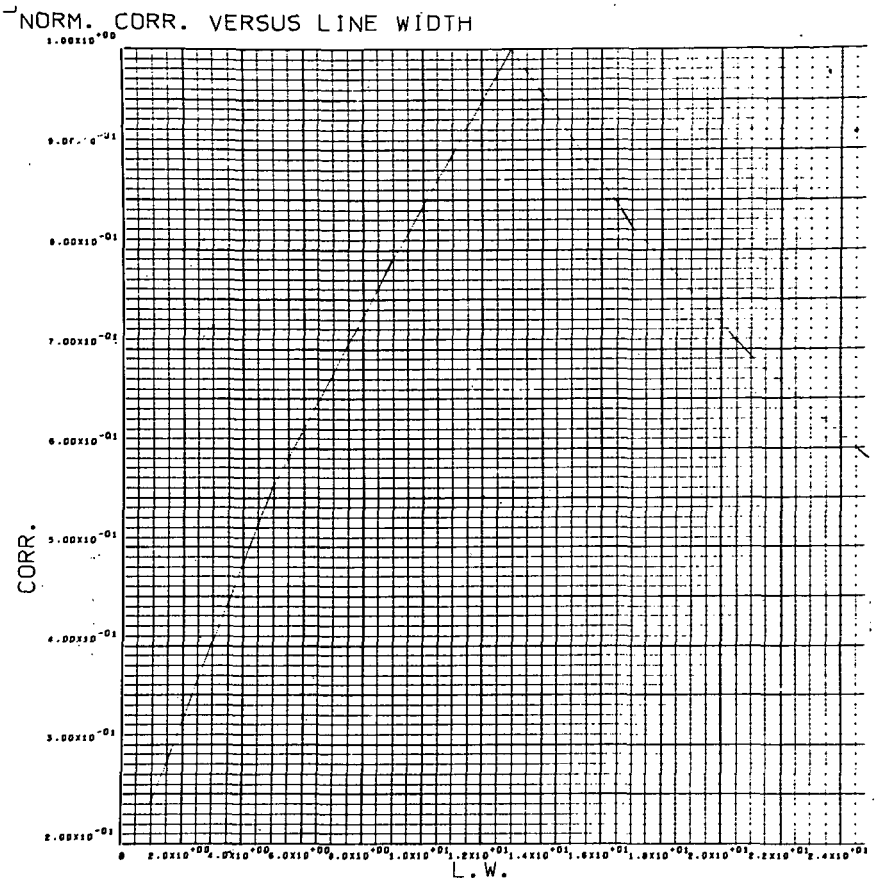


Figure 3.3c Filter Output at Centers of Vertical Lines (Noise-Free Test Pattern)

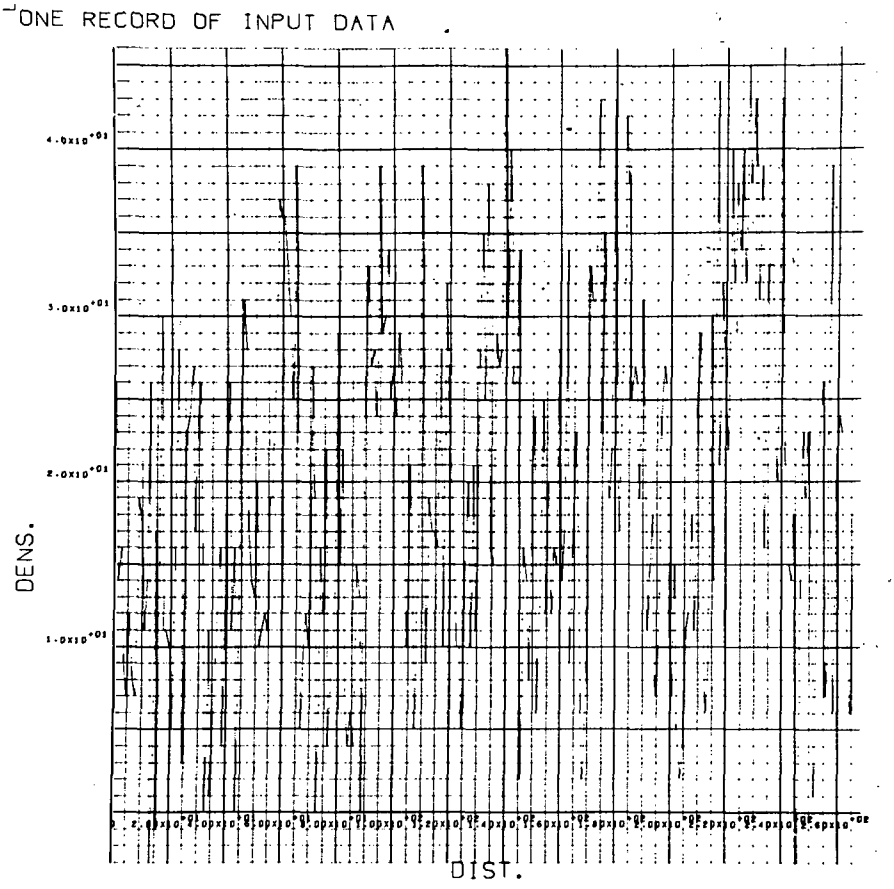


Figure 3.4a One Record of Noisy Test Pattern

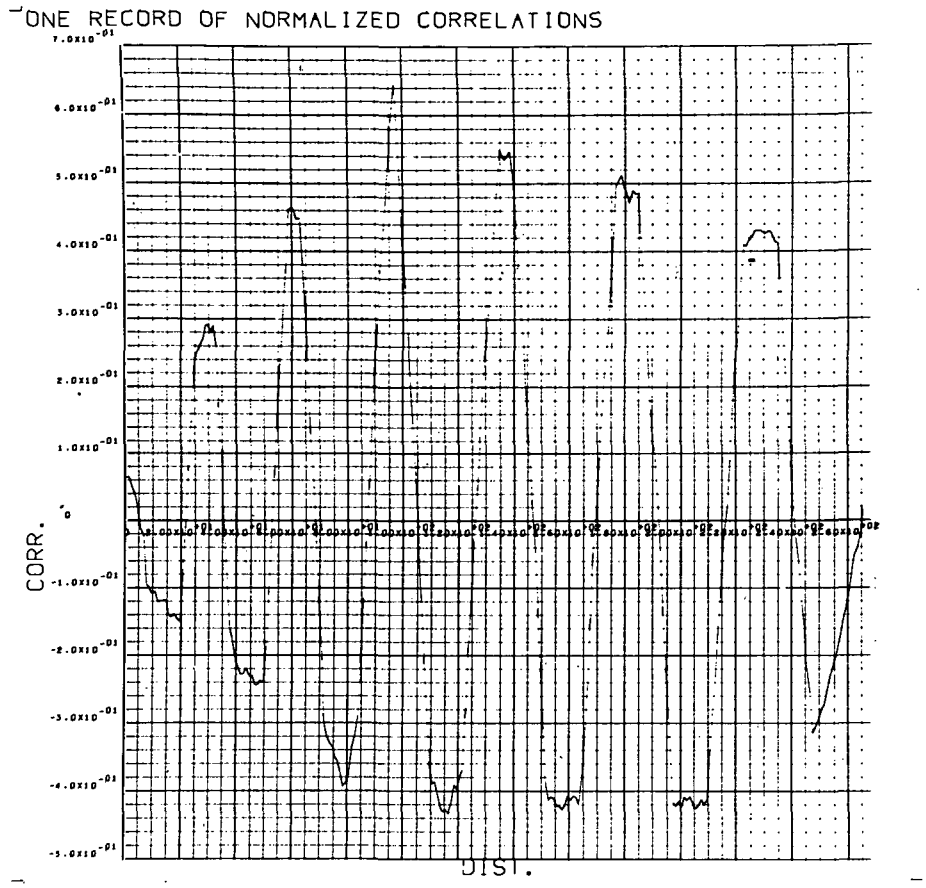


Figure 3.4b Matched Filter Output Corresponding to the Record in Figure 2.4a

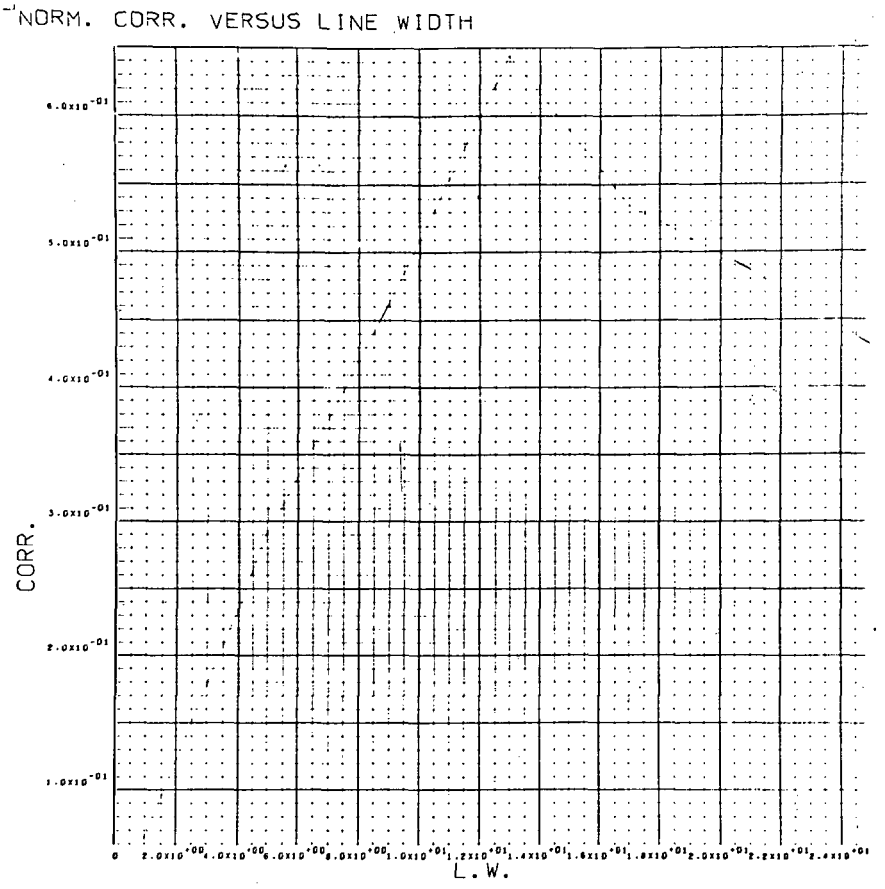


Figure 3.4c Filter Output at Centers of Vertical Lines (Noisy Test Pattern)

```
      SUBROUTINE TOFIL1(NEL,HORFIL,PAR,IX,X,W,Y,W1)
C     THIS SUBROUTINE IMPLEMENTS A -PRODUCT TYPE - SEPARABLE RECURSIVE
C     FILTER WHICH IS LOW PASS IN THE VERTICAL DIRECTION. HORFIL IS AN
C     EXTERNAL ROUTINE WHICH PRODUCES THE DESIRED HORIZONTAL FILTER FROM
C     THE INPUT AND THE HORIZONTAL MOVING SUMS OF THE INPUT.
C     X AND IX SHD BE EQ-NCED. Y,W1 ARE NEEDED ONLY IF THE FILTER OUTPUT
C     IS TO BE NORMALIZED. IN THIS CASE Y AND W1 SHD BE D-NED NEL. PAR(.)
C     ARE PARAMETERS TO BE USED IN HORFIL. PAR(1) IS USED IN RLPFL1, A
C     SUBROUTINE WHICH PRODUCES HORIZONTAL MOVING SUMS OF 2*PAR(1)-1
C     TERMS AT A TIME.
C      INPUT TAPES-- DATA ON UNIT 8 AND DATA+REFLCTIONS ON UNIT 10
C      OUTPUT--- UNIT 11
      LOGICAL NORM
      COMMON /INPAR/NREC,NRECR,ITYPE
      COMMON /VERPAR/LR,NORM
      DIMENSION IX(NEL),X(NEL),W(NEL),Y(1),W1(1),PAR(1)
C     COMMON BLOCK INPAR--PARAMETERS RELATING TO INPUT DATA (COMMON TO
C     CALLING PROGRAM AND THIS ROUTINE)
C     NREC=NUMBER OF INPUT RECORDS, NRECR=NUMBER OF RECORDS REFLECTED A
C    1T THE TOP AND BOTTOM OF THE PICTURE BY SUBROUTINE REFLC. ITYPE=0
C     FOR INTEGER INPUT AND 1 FOR REAL INPUT.
C     COMMON BLOCK VERPAR--PARAMETERS RELATING TO VERTICAL FILTER.
C     2LR-1=VERTICAL FILTER LENGTH. NORM IS .TRUE. IF FILTER OUTPUT IS
C     TO BE NORMALIZED(I.E., DIVIDED BY THE SUM OF SQS. OF DEVIATIONS
C     OF INPUT FROM LOCAL AVERAGE--BOTH THE SUMS AND AVE. ARE OVER A
C     (2LR-1)*(2PAR(1)-1) RECTANGLE CENTERED AT THE POINT UNDER CONS-
C     IDERATION).
C     FIRST FIND THE FIRST RECORD OF VERTICAL SUMS. VERTICAL SUMS ARE
C     STORED IN W.
      SV=2*LR-1
      S=SV*(2.*PAR(1)-1.)
      IRSKP=NRECR-LR+1
      IF(IRSKP.EQ.0) GO TO 20
      DO 10 I=1,IRSKP
10    READ(10)IR,(IX(IEL),IEL=1,NEL)                                    9
20    NREC1=SV
      JREC=0
      DO 40 IEL=1,NEL
40    W(IEL)=0.
      IF(.NOT.NORM) GO TO 42
      DO 44 IEL=1,NEL
44    Y(IEL)=0.
42    CONTINUE
      DO 30 IREC=1,NREC1
      READ(10) IR,(IX(IEL),IEL=1,NEL)                                   40
      IF(ITYPE.EQ.1) GO TO 35
      CALL VADDI(IX,X,W,W,NEL)                                          49
      IF(NORM) CALL VSADDI(IX,X,Y,Y,NEL)                                52
      GO TO 30
35    CALL VADDR(IX,X,W,W,NEL)                                          56
      IF(NORM) CALL VSADDR(IX,X,Y,Y,NEL)                                59
30    CONTINUE
C     FIND MOVING SUMS OF W USING RLPFL1 AND STORE IN X. FIND THE NORM
C     IF NORMALIZATION IS NEEDED. Y AND W ARE HELD UNCHANGED FOR RECUR
C     SION.
```

```
      CALL RLPFL1(W,X,NEL,PAR)                                          64
      IF(.NOT.NORM) GO TO 56
      CALL RLPFL1(Y,W1,NEL,PAR)                                         69
      DO 58 IEL=1,NEL
   58 W1(IEL)=SQRT(W1(IEL)-X(IEL)**2/S)
C     IF NORM=.TRUE. THE NORMS HAVE NOW BEEN FOUND IN W1. HORFIL IS
C     NOW OPERATED ON W AND X TO PRODUCE OUTPUT IN X.                   78
   56 CALL HORFIL(W,X,NEL,PAR)                                          83
      IF(.NOT.NORM) GO TO 57
      DO 59 IEL=1,NEL
      IF(W1(IEL).EQ.0.) GO TO 59
      X(IEL)=X(IEL)/W1(IEL)
   59 CONTINUE
   57 JREC=JREC+1
      WRITE(11) JREC,(X(IEL),IEL=1,NEL)
C     CALL BSFILE AND SKIP NRECR-NR+1 RECORDS ON 10 AND LR RECORDS ON 8 102
      CALL BSFILE(10,1)                                                 109
      DO 60 I=1,IRSKP
   60 READ(10) IR,(IX(IEL),IEL=1,NEL)                                   114
      DO 70 I=1,LR
   70 READ(8) IR,(IX(IEL),IEL=1,NEL)
C     FIND MOVING VERTICAL SUMS RECURSIVELY, OPERATE RLPFL1 AND HORFIL,
C     NORMALIZE THE OUTPUT IF NORM=.TRUE. AND WRITE THE OUTPUT ON 11
C     FOR RECORD NUMBERS 2 THRU NREC-LR+1                               125
      NREC1=2
      NREC2=NREC-LR+1
      NTP1=8
      NTP2=10
      NPASS=1
   80 CONTINUE
      DO 80 I=NREC1,NREC2
      IF(ITYPE.EQ.1) GO TO 85
      READ(NTP1) IR,(IX(IEL),IEL=1,NEL)                                 143
      CALL VADDI(IX,X,W,W,NEL)                                          149
      IF(NORM) CALL VSADDI(IX,X,Y,Y,NEL)                                152
      READ(NTP2) IR,(IX(IEL),IEL=1,NEL)                                 153
      CALL VSUBI(IX,X,W,W,NEL)                                          159
      IF(NORM) CALL VSSUBI(IX,X,Y,Y,NEL)                                162
      GO TO 99
   85 CONTINUE
      READ(NTP1) IR,(X(IEL),IEL=1,NEL)                                  165
      CALL VADDR(IX,X,W,W,NEL)                                          171
      IF(NORM) CALL VSADDR(IX,X,Y,Y,NEL)                                174
      READ(NTP2) IR,(X(IEL),IEL=1,NEL)                                  175
      CALL VSUBR(IX,X,W,W,NEL)                                          181
      IF(NORM) CALL VSSUBR(IX,X,Y,Y,NEL)                                184
   99 CONTINUE
      CALL RLPFL1(W,X,NEL,PAR)                                          187
      IF(.NOT.NORM) GO TO 96
      CALL RLPFL1(Y,W1,NEL,PAR)                                         192
      DO 98 IEL=1,NEL
   98 W1(IEL)=SQRT(W1(IEL)-X(IEL)**2/S)
C     NORMS HAVE BEEN COMPUTED IN W1 IF NORM=.TRUE.                     200
   96 CALL HORFIL(W,X,NEL,PAR)                                          204
      IF(.NOT.NORM) GO TO 97
      DO 91 IEL=1,NEL
```

```
      IF(W1(IEL).EQ.0.) GO TO 91
      X(IEL)=X(IEL)/W1(IEL)
91    CONTINUE
97    JREC=JREC+1
      WRITE(11) JREC,(X(IEL),IEL=1,NEL)                                  221
80    CONTINUE
C     AT THE END OF FIRST PASS THRU LOOP 80 UNIT 8 HAS FINISHED READING
C     THE LAST RECORD IN THE FILE AND UNIT 10 IS READY TO READ THE NREC
C     -2*LR+2TH RECORD IN THE ORIGINAL PICTURE(I.E.THE NRECR+NREC-2*LR+
C     2TH RECORD OF THE FILE ON UNIT 10).
      NPASS=NPASS+1
      IF(NPASS.GT.2) GO TO 110
C     DURING THE SECOND PASS THRU LOOP 80 THE ROLES OF 8 AND 10 ARE
C     INTERCHANGED. TO PREPARE FOR THIS BSFILE AND SKIP NREC-2*LR+1
C     RECORDS ON 8 AND SKIP(NO REWIND) 2*LR-1 RECORDS ON 10.
      CALL BSFILE(8,1)                                                   233
      NREC1=2*LR-1
      DO 130 I=1,NREC1
130   READ(10) IR,(IX(IEL),IEL=1,NEL)                                    239
      NREC1=NREC-2*LR+1
      DO 120 I=1,NREC1
120   READ(8) IR,(IX(IEL),IEL=1,NEL)                                     251
      NTP1=10
      NTP2=8
      NREC1=NREC-LR+2
      NREC2=NREC
C     GO TO 140 AND COMPUTE AND WRITE THE OUTPUTS FOR RECORD NUMBERS
C     NREC-LR+2 THRU NREC.
      GO TO 140
110   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VACDI(IX,X,Y,W,NEL)
      DIMENSION IX(NEL),X(NEL),Y(NEL),W(NEL)
      DO 10 IEL=1,NEL
10    W(IEL)=Y(IEL)+FLOAT(IX(IEL))
      RETURN
      ENTRY VSACDI(IX,X,Y,W,NEL)
      DO 20 IEL=1,NEL
20    W(IEL)=Y(IEL)+FLOAT(IX(IEL))**2
      RETURN
      ENTRY VSUBI(IX,X,Y,W,NEL)
      DO 30 IEL=1,NEL
30    W(IEL)=Y(IEL)-FLOAT(IX(IEL))
      RETURN
      ENTRY VSSUBI(IX,X,Y,W,NEL)
      DO 40 IEL=1,NEL
40    W(IEL)=Y(IEL)-FLOAT(IX(IEL))**2
      RETURN
      ENTRY VADDR(IX,X,Y,W,NEL)
      DO 50 IEL=1,NEL
50    W(IEL)=Y(IEL)+X(IEL)
      RETURN
      ENTRY VSADDR(IX,X,Y,W,NEL)
      DO 60 IEL=1,NEL
60    W(IEL)=Y(IEL)+X(IEL)**2
      RETURN
      ENTRY VSUBR(IX,X,Y,W,NEL)
      DO 70 IEL=1,NEL
70    W(IEL)=Y(IEL)-X(IEL)
      RETURN
      ENTRY VSSUBR(IX,X,Y,W,NEL)
      DO 80 IEL=1,NEL
80    W(IEL)=Y(IEL)-X(IEL)**2
      RETURN
      END
```

```
      SUBROUTINE RLPFL1(X,Y,NEL,PAR)
C     THIS PROGRAM GENERATES Y(IEL)=X(IEL-PAR(1)+1)+...+X(IEL+PAR(1)-1)
      DIMENSION X(NEL),Y(NEL),PAR(1)
      LC=PAR(1)
      Y(1)=0.
      L2=LC*2-1
      L1=LC-1
      NEL2=NEL*2
      DO 10 I=1,L2
      J=I-L1
      IF(J.LE.0) J=2-J
      IF(J.GT.NEL) J=NEL2-J
10    Y(1)=Y(1)+X(J)
C     NOW COMPUTE  Y(I) USING Y(I)=X(I+LC-1)-X(I-LC)+Y(I-1)
      DO 20 I=2,NEL
      J1=I-LC
      J2=I+LC-1
      IF(J1.LE.0) J1=2-J1
      IF(J2.GT.NEL) J2=NEL2-J2
20    Y(I)=Y(I-1)-X(J1)+X(J2)
      RETURN
      END
```

```
      SUBROUTINE RHPFL1(X,Y,NEL,PAR)
C     THIS SUBROUTINE PRODUCES THE OUTPUT OF A HIGHPASS FILTER IN Y. X
C     IS THE INPUT. Y IS THE OUTPUT OF RLPFL1 AT THE BEGINNING OF CALL.
C     2PAR(1)-1 IS THE FILTER LENGTH. THE EFFECT OF THE FILTER IS TO
C     SUBTRACT MOVING AVERAGES.
      DIMENSION X(NEL),Y(NEL),PAR(1)
      S=2.*PAR(1)-1.
      DO 10 IEL=1,NEL
10    Y(IEL)=X(IEL)-Y(IEL)/S
      RETURN
      END
```

```
      SUBROUTINE RBPFL1(X,Y,NEL,PAR)
C     THIS SUBROUTINE PRODUCES THE OUTPUT OF A BANDPASS FILTER REALIZED
C     AS THE DIFFERENCE BETWEEN TWO LP FILTERS. PAR SHD BE D-NED NEL+2.
C     2PAR(1)-1 AND 2PAR(2)-1 ARE THE LENGTHS OF THE LP FILTERS. PAR(1)
C     SHD BE .GT. PAR(2). AT THE BEGINNING OF CALL X IS THE INPUT RECOR
C     AND Y IS THE OUTPUT OF RLPFL1 WITH PAR(1) AS THE LENGTH PARAMETER
C     AT THE END OF CALL X IS UNCHANGED BUT Y IS THE BP FILTER OUTPUT.
      DIMENSION X(NEL),Y(NEL),PAR(1)
      S1=2.*PAR(1)-1.
      S2=2.*PAR(2)-1.
      CALL RLPFL1(X,PAR(3),NEL,PAR(2))                              2
      DO 10 IEL=1,NEL
10    Y(IEL)=PAR(IEL+2)/S2-Y(IEL)/S1
      RETURN
      END
```

REFERENCES

1. Seltzer, R. H., "The Use of Computers to Improve Biomedical Image Quality," AFIPS Conference Proceedings, Volume 33, Part I, 1968 (Fall Joint Computer Conference), pp. 817-834.

2. Kuo, F. F. and Kaiser, J. F., System Analysis by Digital Computers, John Wiley and Sons, Inc., 1966, pp. 218-285.

3. Papoulis, A., Systems and Transforms with Applications in Optics, McGraw Hill Publishing Co., 1968.

4. Ormsby, J. F. A., "Design of Numerical Filters with Application to Missile Data Processing," J. ACM, Volume 8, July 1961, pp. 440-466.

5. Nathan, R., Digital Video-Data Handling NASA Technical Report No. 32-877, Jet Propulsion Laboratory, Calif., Inst. of Tech., January 1966.

6. Cooley, J. W., Lewis, P. A. W. and Welch, P. D., "The Fast Fourier Transform Algorithm and Its Applications" IBM Research Paper RC-1743, February 9, 1967.

7. Helms, H. D., "Fast Fourier Transform Method of Computing Difference Equations and Simulating Filters," IEEE Trans. on Audio and Electroacoustics, Volume AU-15, No. 2, June 1967.

8. Rabiner, L. R., Gold, B. and McGonegal, C. A., "An Approach to the Approximation Problem for Nonrecursive Digital Filters," IEEE Trans. on Audio and Electroacoustics, Volume AU-18, No. 2, June 1970, pp. 83-106.

9. Turin, G. L., "An Introduction to Matched Filters," IRE Trans, on Information Theory, June 1960, pp. 311-329.

10. Davenport, W. B. and Root, W. L., An Introduction to the Theory of Random Signals and Noise," p. 224, McGraw Hill Publishing Co., New York, New York, 1958.

11. Cook, C. E. and Bernfeld, M., Radar Signals, Academic Press, New York, 1967.

12. Andrews, H. C., et al, Computer Techniques in Image Processing, Academic Press, New York, 1970, pp. 55-71.

13. Brown, W. M., Analysis of Linear Time-Invariant Systems, McGraw Hill Publishing Company, New York, 1963, pp. 244-245.

14. Rosenfeld, A., Picture Processing by Computer, Academic Press, New York, 1969.

15. Vander Lugt, A., "Signal Detection by Complex Spatial Filtering," IEEE Trans. on Information Theory, Volume IT-10, No. 2, April 1964, pp. 139-145.

16. Stockham, T. G., "High-Speed Convolution and Correlation," Proc. of 1966 Spring Joing Computer Conference, AFIPS Proc., Volume 28, Washington, D.C., Spartan, 1966, pp. 229-233.

17. Helms, H. D., "Fast Fourier Transform Method for Computing Difference Equations and Simulating Filters," IEEE Trans. on Audio and Electroacoustics, Volume AU-15, No. 2, June 1967.

18. Rabiner, L. R. and Schafer, R. W., "Recursive and Nonrecursive Realizations of Digital Filters Designed by Frequency Sampling Techniques," IEEE Trans. on Audio and Electroacoustics, Volume AU-19, No. 3, September 1971, pp. 200-207.

19. Tou, J. T., Digital and Sampled Data Control Systems, McGraw Hill, 1959.

20. Barnea, D. I. and Silverman, H. F., "A Class of Algorithms for Fast Digital Image Registration," IEEE Trans. on Computers, Volume C-21, No. 2, February 1972, pp. 179-186.

21. Rosenfeld, A., et al, "Edge and Curve Detection: Further Experiments," IEEE Trans. on Computers, Volume C-21, No. 7, July 1972.

☆U.S. GOVERNMENT PRINTING OFFICE:1973-714-531/2171

Page Intentionally Left Blank

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

SPECIAL FOURTH-CLASS RATE BOOK

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
451

U.S.MAIL

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546